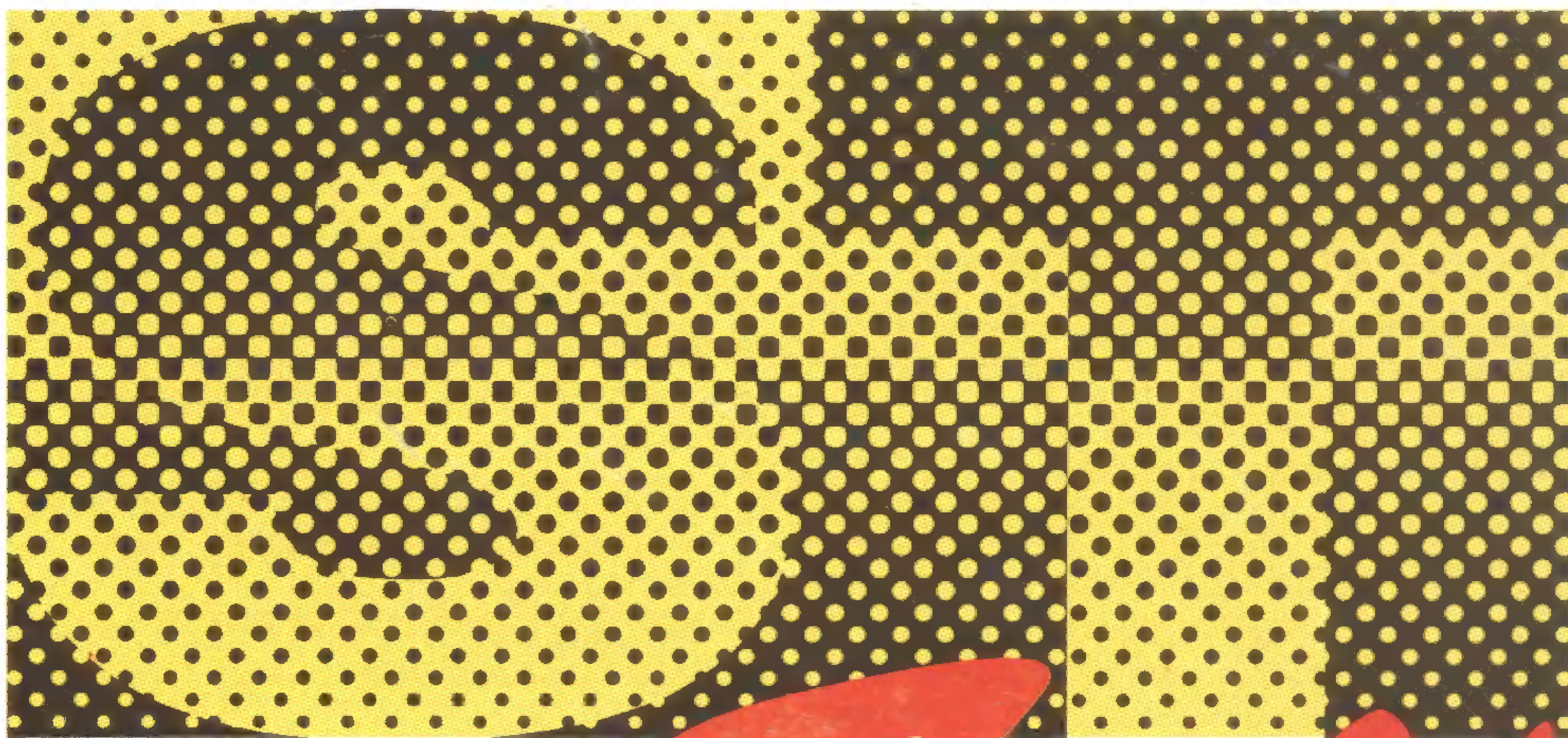
ATARI ST · ATARI STE · ATARI TT

2

CENA 15 000 zł

1993 · ROK 2, NR 2(4)

POPULARNY MAGAZYN
UŻYTKOWNIKÓW
KOMPUTERÓW ATARI ST



INDEKS 37590X

Fan

Yamaha

EMULATORY WYSOKIEJ ROZDZIELCZOŚCI

BOOTSECTOR

EDYTOR MIDI

OD REDAKTORA

PIRAT !

Studio Atari ze Szczecina postawiło na polskie oprogramowanie. Powstają w nim nasze rodzime programy użytkowe. Studio to jest też dystrybutorem innych polskich programów. Kilka tygodni temu, przyjechał do Warszawy na giełdę komputerową „kolega" z Gdańska i próbował sprzedać piracką kopię, jak sam nazwał „świetnego polskiego programu FBI". Dzięki temu, że jestem na giełdzie warszawskiej znany nie udało mu się sprzedać ani jednej kopi.

Drogi „kolego" z Gdańska! Na łamach „ST-FANa" ostrzegam Cię rzetelnie, że jeżeli nie zaprzestaniesz sprzedawać moich programów nielegalnie to zaskarżę Cię z powództwa cywilnego do sądu. Twoje nazwisko i adres są mi dobrze znane. Otrzymasz jeszcze ode mnie stosowny list. Tak na marginesie, to są już na ukończeniu trzy kolejne polskie programy! Tytuły zdradzę dopiero w kolejnym numerze „ST-FANa".

Drodzy Czytelnicy nie kupujcie pirackich kopi!. Nasze oprogramowanie nie jest drogie. Pamiętajcie, że kupując od nas legalny program otrzymujecie ładne opakowanie i instrukcję ubsługi. Kolejne wersje programu otrzymujecie za niewielką dopłatą. Miejcie też na uwadze to, że napisanie programu wymaga sporo wysiłku i pracy, a nikt nie chce pracować za darmo! Jeżeli zwalczać będziesz piratów , Sam się przekonasz, jak dużo polskich programów ukaże się do końca roku na naszym rynku.

CHOCHLIK

Życie redaktora pisma nie jest usłane różami. Oprócz pochwał (których jest mało), zbiera on słowa krytyki, zażalenia i przyjmuje na siebie odpowiedzialność za błędy i niedopatrzenia. A wszystkiemu winien jest nie redaktor tylko CHOCHLIK redakcyjny, który postanowił w naszej redakcji przezimować i przy okazji płata nam figle. Przykładem jego kolejnych figli jest poprzestawianie stron w numerze 1/93, a konkretnie wydrukował dwie takie same strony. Bardzo przepraszam za to Czytelników i Kolegów z Lipna. Mam dużą nadzieję, że już nie długo wraz z nadejściem wiosny wyprowadzi się od nas.

DZIĘKUJĘ !

Chciałbym serdecznie podziękować w imieniu całego zespołu redakcyjnego i własnym, za miłe i serdeczne życzenia noworoczne. Szczególne słowa podziękowania należą się Karolinie Chełminiak z Piły, która na tą okoliczność napisała specjalny program z życzeniami dla „ST-FANa". Szkoda, że nie mogę Wam zaprezentować tego programu. Bardzo ładna grafika i dźwięk. W trakcie trwania kolędy ukazują się jej słowa, a na zakończenie dodatkowo ładny napis z życzeniami. Dziękujemy!

ADRES

Większość z Was zauważyła zmianę adresu redakcji. Ci którzy zmiany nie zauważyli pisali na stary adres. Teoretycznie nie powinno być problemu, ponieważ w poprzednim miejscu mieliśmy obiecane, że korespondencja do nas napływająca będzie odbierana i nam przekazywana. W praktyce wyszło zupełnie inaczej. Korespondencja była nam przekazywana tylko przez pierwsze kilka tygodni, a obecnie nie przyjmują jej wcale. Dlatego otrzymujecie zwroty. Do tego wszystkiego beszczelnie kłamią, że żadne listy do nas nie napływają. Dopiero od naszych rozgoryczonych Czytelników dowiedziałem się o zaistniałej sytuacji. Za powstałe nieporozumienie bardzo przepraszam.

REDAKTOR

Podaję obecny dokładny adres:

WYDAWNICTWO „ST-FAN"
71-421 SZCZECIN
al. WYZWOLENIA 103/16
tel.: (0-91) 22 34 37

SPIS TREŚCI

NR 2 (4). 1993

POPULARNY MAGAZYN UŻYTKOWNIKÓW KOMPUTERÓW ATARI ST



INDEX 37590X

REDAKCJA
ST-Fan
71-421 Szczecin,
al. Wyzwolenia 103/16
tel.: (0-91) 22 34 37

Andrzej Sobieraj
(redaktor naczelny)
Sławomir Barcz
Jacek Kulijewicz
Przemysław Mazurek
Wiesław Racki
(oprac. graf. i skład)

WYDAWCA
WYDAWNICTWO



DRUK
P.P.H. ZAPOL
Sz-n, ul. Piastów 42

INFORMATOR NOWOŚCI *i nie tylko*

2,8 MB

STACJA DYSKIETEK ED

Super-gęstą stację 3,5" ED czyli 2,88 MB po sformatowaniu, można zainstalować w Atari, wymagany TOS od 2.00. Zestaw składa się z modułu ED-Kit 3, stacji TEAC FD235J-ED i kontrolera AJAX. Dodatkiem jest program E-Format, całość kosztuje około 400DM.

DIRECT TO DISC

PROGRAMY NA FALCONA

Zaczynają pojawiać się pierwsze programy wykorzystujące wspaniałe możliwości Falcona. Ukazał się mianowicie Direct To Disc (D2D), program do cyfrowego zapisu i obróbki dzwięku. Potrafi zapisywać z jakością lepszą od kompaktu na wewnętrzny lub zewnętrzny twardy dysk w czasie rzeczywistym..

Z efektów oferuje echo, chorus, reverb i flange (też na żywo). Zużywa 10Mb na zapis jednej minuty .Program działa również na Mega ST(E) potrzebując jednak min. 150Mb twardego dysku i 16 bitowego samplera. (L)

FALCON I IBM PC

KUPUJCIE FALCONA

Większość z Was, na pewno czytała dane techniczne Falcona 030 i wiedzą że komputer ten przewyższa Atari ST dość znacznie. Jak jednak prezentuje się na tle sprzętu zgodnego z IBM PC wie niewielu a warto to wiedzieć by nie dać się „zauroczyć" pecetami.

Otóż szybkość procesora MC 68030 16Mhz montowanego w nowym Atari wynosi 3.85 MIPS (milionów operacji na sekundę). Odpowiada to komputerowi 386SX 25Mhz po stronie PC. Jednak szybkość całego systemu jest taka jak 386DX 33Mhz - większa szybkość zegara potrzebna jest, aby zrekompensować straty spowodowane przez nakładki (MS Windows) na system operacyjny (MS DOS), który z kolei trzeba wczytać na „goły" komputer.Uff!!.

Nie należy też zapominać o procesorze DSP56K, który nie służy tylko do obsługi cyfrowego dzwięku i kolorowego wideo lecz może także wspomagać główny procesor w innych zadaniach. Przy jego efektywności 16 MIPS daje nam to szybkość pracy identyczną jak w dużych i drogich komputerach PC 486/33Mhz !! Jeżeli dołożymy do tego wspaniałą kartę graficzną Falcona, bardzo szybką grafikę, możliwość wczytywania i obróbki obrazów wideo bez potrzeby dokupywania drogich rozszerzeń (karta do PC o takich możliwościach kosztuje kilka tysięcy dolarów...) a także dzwięk o jakości płyty kompaktowej (kolejny tysiąc dolarów po stronie PC) to otrzymamy sprzęt o bardzo dużych możliwościach i niskiej cenie.

Wliczając jeszcze dużą ilość interfejsów Falcona i twardy dysk do niego okaże się, że aby otrzymać peceta o możliwościach Atari musimy wydać około 5000 $.

Falcon z 4Mb pamięci i twardym dyskiem 65Mb kosztuje 1400$. System operacyjny Atari, MultiTOS 4.00, oferuje pełną wielozadaniowość , o którą w przypadku PC i systemu Windows można pokusić się tylko na bardzo szybkich komputerach, a i na nich zdarzają się czasem wpadki kończące się zwykle zawieszeniem maszyny i utratą cennych danych. (L)

Na podstawie ST FORMAT 11/92

DTP

KONKURNCJA CALAMUSA ?

Ukazała się nowa wersja programu Timeworks Publisher 2 będącego popularnym programem do DTP. Z oferowanych nowości wymienić należy obsługę nowych formatów graficznych (TIFF, EPS, PCX, MacPaint) i pełną zgodność z wesją 1.12 na ST i Timeworks 2 na IBM PC. Wymagania to minimum 1Mb pamięci i dwustronna stacja dysków. (L)

BOOK

PRZENOŚNE ATARI

Firma ATARI wprowadziła na rynek przenośną wersję ATARI ST. ST BOOK wyposażony jest w ciekłokrystaliczny ekran rozdzielczości 640*400 punktów doskonałej jakości i twardy dysk o standardowej pojemności 40MB. W ROMie komputera oprócz TOSu znajduje się oprogramowanie komunikacyjne pozwalające na komunikację z PC, ST i Portfolio.

Ten notebook posiada te same złącza, które zastosowano w domowej wersji komputera. Przy rozmiarach rozłożonego zeszytu, wadze poniżej 2 kg, pozwala na około 10 godzinną pracę na siedmiu bateriach R6. Cena około 3.000 DM. (L)

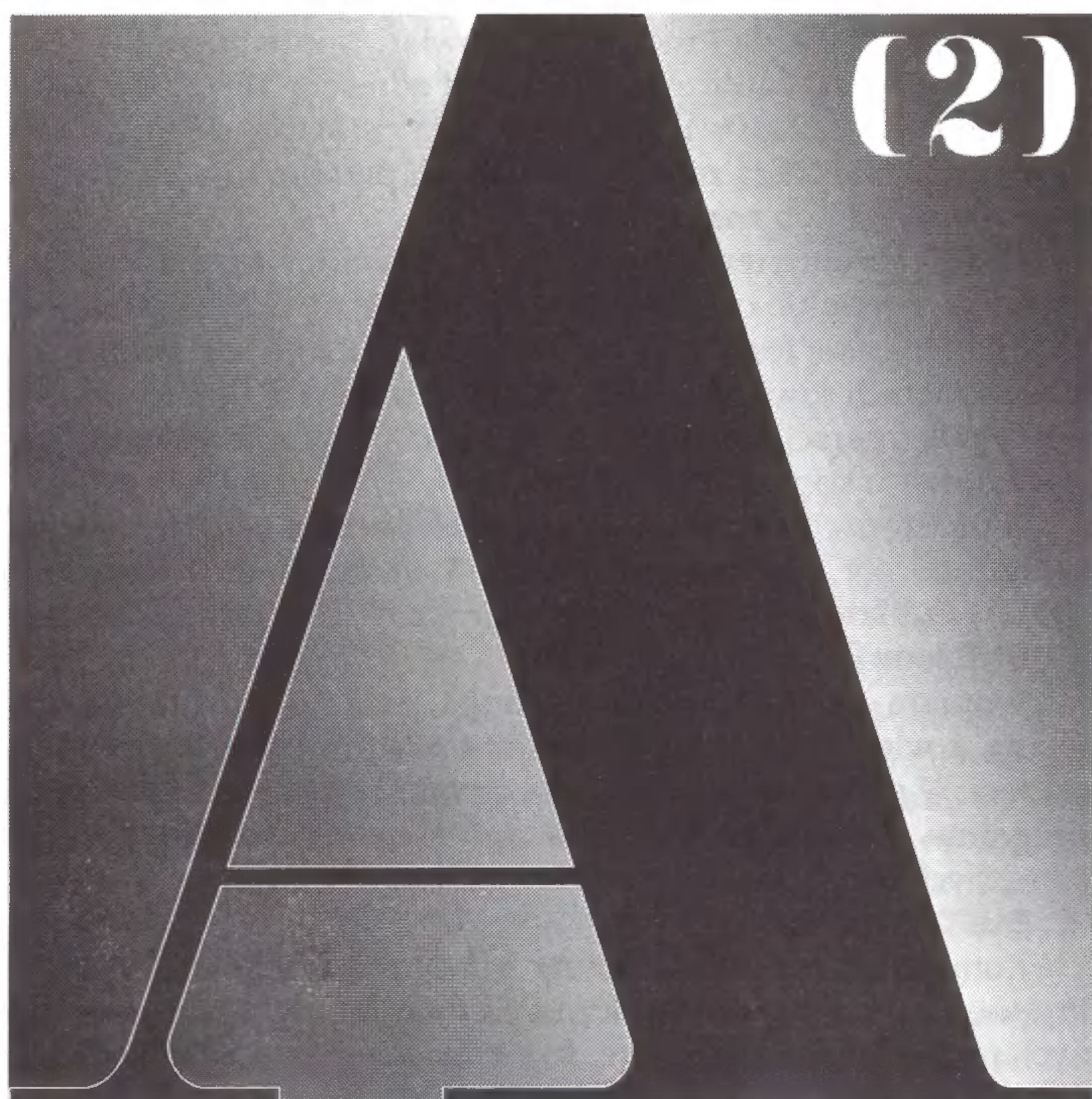
TOS 2.06

ODMŁADZANIE ATARI ST

W Wielkiej Brytani popularną ostatnio usługą jest montaż nowych kości ROM zawierających TOS 2.06 (notabene montowany wcześniej w ATARI MEGA STE). Nowy system operacyjny lepiej współpracuje z twardym dyskiem, lepiej operuje plikami, pozwala na tworzenie dowolnych ikon i konfigurację biurka wedle upodobań.

Nowy TOS nie koliduje z innymi kartami zamontowanymi wewnątrz ST, a w razie potrzeby można zpowrotem włączyć stary system zamontowanym przełącznikiem. (L)

PROGRAMOWANIE NA ATARI ST



ASEMBLER

Mamy już pewne względne pojęcie o sposobie w jaki komputer przechowuje informacje w pamięci w postaci kodu lub danych. Poniższy odcinek asemblera poświęcimy analizie kodów źródłowych 2 programów. Aby lepiej zrozumieć ich działanie, będą one bezpośrednio operowały na pamięci ekranu. Dzięki temu efekty ich pracy będą od razu widoczne. Przejdźmy do pierwszego listingu.

```
*instrukcje wpisujemy po uprzednim tabulatorze *
clr.l -(sp)
move.w #$20,-(sp)
trap #1
addq #6,sp ;supervisor

move.l #8000,d0
lea $F8000,a0 loop:
move.l #%11111111111111111111111111111111,(a0)+
dbf d0,loop

clr.l -(sp)
move.w #1,-(sp)
trap #1
addq #6,sp

clr.l -(sp)
trap #1
```

Po wpisaniu tekstu, zaasemblerowaniu go do pamięci i uruchomieniu komputer wypełni ekran kolorem 15. Będzie on następnie czekał na wciśnięcie dowolnego klawisza po czym powróci do edytora. Pierwsze 4 instrukcje wywołują procedurę systemową, która powo duje przejście procesora do pracy w trybie nadzorcy. W trybie tym procesor może adresować całą dostępną pamięć RAM. W naszym przypadku jest to konieczne, gdyż będziemy się odwoływać bezpośrednio do pamięci ekranu. Kolejna instrukcja wpisuje liczbę 8000 do rejestru danych d0, „l" wskazuje, że liczba ta jest zapisywana w 4 bajtach. Pojawia się tu nowy mnemonik:

LEA - umieszcza w określonym rejestrze adresowym adres efektywny. Zmienia on wszystkie 4 bajty rejestru adresowego. Za zwyczaj wykorzystywany jest w programach relokowalnych, tj. takich, które będą działały niezależnie od ich położenia w pamięci. Przy pomocy tej instrukcji możemy dodać stałą do rejestru adresowego bez zmiany flag X,N,Z,V,C rejestru statusowego:

LEA 10(a0),a0

LEA $F8000,a0 powoduje, że do rejestru a0 zostanie wpisana wartość $F8000. ($F8000 jest adresem początku pamięci ekranu fizycznego). Ten sam efekt możemy uzyskać wykonując; MOVEA #$F8000,a0

Następnie w programie znajduje się etykieta LOOP. Etykiety są to wskaźniki do odpowiednich fragmentów kodu. Dzięki nim wykonując skoki nie musimy odwoływać się bezpośrednio do adresów efektywnych.

MOVE.L #%11111111111111111111111111111111,(a0)+

Znak „%" przy atrybucie instrukcji oznacza, że będzie on zapisany w postaci binarnej. Pod adres znajdujący się w rejestrze a0 zostanie wpisane długie słowo #%11111111111111111111111111111111, po czym do rejestru a0 dodawana jest liczba 4.

DBF rejestr danych,etykieta

Pierwszy rodzaj pętli warunkowych. Instrukcji typu DB. jest kilka, związane są one z ustawieniem flag warunkowych rejestru statusowego, w efekcie z różnymi warunkami opuszczenia pętli. Instrukcja DBF działa w następujący sposób: Od rejestru danych odejmowane jest 1, procesor sprawdza czy rejestr danych zawiera wartość 0 (czy flaga Z=1), jeżeli nie to wykonuje skok do etykiety. W przeciwnym razie (jeżeli flaga Z=1) licznik programu pobiera instrukcję następującą po DBF. Mechanizm ten wykorzystany jest przy tworzeniu pętli. W poniższym przykładzie kod pętli zostanie wykonany 8001 razy:

```
move.l #8000,d0
loop:
kod do wykonanania w pętli
DBF d0,loop
```

Ostatnie dwa bloki listingu są wywołaniem procedur systemowych oczekiwania na klawisz oraz wyjścia z programu. Teraz zajmiemy się analizą listingu drugiego programu. Jest on znacznie bardziej rozbudowany, zawiera skok do podprogramu, instrukcję porównania oraz kolejną instrukcję warunkową. A oto jego listing:

```
*
clr.l -(sp)
move.w #$20,-(sp)
trap #1
addq #6,sp

clr.l -(sp)
move.l #text,-(sp)
move.w #9,-(sp)
trap #1
add #10,sp
loop1:
lea $F8000,a0
rept 150
bsr scroll
adda.l #160,a0
endr
cmp.b #185,$FFFC02
bne loop1

clr.l -(sp)
trap #1 *** * Wyjaśnienie struktury ekranu znajduje się w osobnym artykule. *** scroll:
move.b (a0),bufor
move.b 7(a0),(a0)
move.b 8(a0),7(a0)
move.b 15(a0),8(a0)
move.b 16(a0),15(a0)
```

```
move.b 23(a0),16(a0)
move.b 24(a0),23(a0)
move.b 31(a0),24(a0)
move.b 32(a0),31(a0)
move.b 39(a0),32(a0)
move.b 40(a0),39(a0)
move.b 47(a0),40(a0)
move.b 48(a0),47(a0)
move.b 55(a0),48(a0)
move.b 56(a0),55(a0)
move.b 63(a0),56(a0)
move.b 64(a0),63(a0)
move.b 71(a0),64(a0)
move.b 72(a0),71(a0)
move.b 79(a0),72(a0)
move.b 80(a0),79(a0)
move.b 87(a0),80(a0)
move.b 88(a0),87(a0)
move.b 95(a0),88(a0)
move.b 96(a0),95(a0)
move.b 103(a0),96(a0)
move.b 104(a0),103(a0)
move.b 111(a0),104(a0)
move.b 112(a0),111(a0)
move.b 119(a0),112(a0)
move.b 120(a0),119(a0)
move.b 127(a0),120(a0)
move.b 128(a0),127(a0)
move.b 135(a0),128(a0)
move.b 136(a0),135(a0)
move.b 143(a0),136(a0)
move.b 144(a0),143(a0)
move.b 151(a0),144(a0)
move.b 152(a0),151(a0)
move.b bufor,152(a0)
rts text: dc.b 'Demonstracja scrolingu na 1 bitplanie',0 bufor: dc.b 0
```

Powyższy program drukuje na ekranie tekst, a następnie przesuwa w lewo 150 lini 1 bitplanu co 8 punktów do momentu wciśnięcia spacji. Pierwsze dwa wywołania to odpowiednio: przejście do trybu nadzorcy oraz wyświetlenie tekstu na ekranie. Natępnie zbudowana jest pętla pomiędzy etykietą LOOP1 a instrukcją BNE. Omówmy najpierw jednak kod wewnątrz pętli. LEA robi dokładnie to samo co w poprzednim przykładzie.

REPT 2
kod do powtórzenia np.: MOVE.L #0,d0
ENDR

Taka konstrukcja spowoduje, że podczas kompilowania programu ' kod do powtórzenia' zostanie 2 razy powtórzony co będzie jednoznaczne zapisowi:

MOVE.L #0,d0
MOVE.L #0,d0

W środku tej pętli znajdują się dwie instrukcje, wywołanie podprogramu oraz dodanie wartości do rejestru adresowego. Przyjrzyjmy się pierwszej z nich.

BSR scroll

Skok do podprogramu, adres instrukcji po nim następującej jest zapisywany na stos, po czym wykonywanie programu rozpoczyna się od podanej etykiety. Przesunięcie między instrukcją skoku a etykietą nie może przekroczyć 65 kB. Kod podprogramu musi być zakończony instrukcją RTS, która pobiera ze stosu zapisany tam adres i przenosi go do licznika programu (PC).

ADDA.L #160,a0

Dodaje 160 bajtów do adresu znajdującego sie w rejestrze adresowym a0. Jest to dokładnie liczba bajtów odpowiadająca jednej lini ekranu. W naszym przypadku przesunięcie 1 bitplanu w lewo oraz przejście do następnej lini zostanie wykonane 150 razy.

CMP.B #185,$FFFC02

Porównuje zawartość rejestru danych klawiatury z wartością 185 (scankod spacji). Precyzyjnie mówiąc, porównując procesor odejmuje 185 od zawartości bajtu pod adresem $FFFC02 i odpowiednio ustawia flagi warunkowe w rejestrze statusowym. Żaden z operandów nie ulega zmianie. W ramach przypomnienia wypiszę wszystkie flagi i ich znaczenie:

C - bit przeniesienia. Może zostać ustawiony przy operacjach arytmetycznych lub obrotach. V - bit nadmiaru. Gdy wynik operacji arytmetycznej jest pozazakresowy. Z - bit zero. Ustawiany gdy wynik operacji równa się 0. N - bit mniejszości od zera. Ustawiany gdy wynik operacji jest mniejszy od zera. X - bit rozszerzenia. Zazwyczaj jest on kopią C, lecz nie zawsze.

W zależności od wyniku operacji arytmetycznej dokonywanej przez procesor, flagi te są ustawiane. W oparciu o nie funkcjonują skoki warunkowe.

BNE loop1

Program „skacze" do etykiety loop1, jeżeli flaga Z nie jest ustawiona, a więc jeżeli wynik ostatniej op. arytmetycznej był różny od zera. W przeciwnym razie wywoływana jest procedura wyścia z programu. Na samym końcu listingu znajduje się blok definicji zmiennych programu.

BUFOR: dc.b 0

Kompilator rezerwuje obszar 1 bajtu dla zmiennej BUFOR. Proponuję przeprowadzić liczne eksperymenty na zamieszczonych przykładach i zapraszam do lektury kolejnego odcinka kursu asemblera.

(Toxical dream)

RECENZJA

CALAMUS ver. 1. 09N
Podręcznik Użytkownika
autor Marian Olejnik

Nakładem warszawskiej Agencji Wydawniczej M&M ukazało się sporo książek omawiających najpopularniejsze programy na ATARI ST. Calamus-Podręcznik Użytkownika , od samego początku robi dobre wrażenie i budzi zaufanie. Jest bardzo starannie wydany. Sztywne, kolorowe okładki, dobrej jakości papier i nienaganny druk idą w parze z treścią.

Podręcznik zawiera dokładny opis programu na 178 stronach. Jest napisany jasno i językiem zrozumiałym dla wszystkich. Autor zaczyna swój poradnik, że tak powiem od początku. Najpierw omawia wymagania sprzętowe, instalację programu, a póżniej, przechodzi metodą „krok po kroku" do omawiania projektowania dwustronicowego dokumentu zawierającego słowa, ilustracje i proste elementy projektowania graficznego, wykorzystywane zwykle w czasie składu drukarskiego.

W kolejnych rozdziałach podręcznika opisuje szczegółowo wszystkie narzędzia, będące w Twojej dyspozycji. Zgodnie z sugestią autora, poradnik należy czytać bezpośrednio przy komputerze, ponieważ wiele zagadnień popartych jest przykładami.

Z pewnością nie jest to lektura na jeden wieczór, ponieważ Calamus jest złożonym programem, ale po uważnym przeczytaniu poradnika i wykonaniu odpowiednio wielu doświadczeń, każdy z Was bez obaw i z rozmachem potrafić będzie bez trudu obsłużyć ten najlepszy program z dziedziny DTP.

Jedynym błędem, czy niedopatrzeniem jest to, że autor poradnika często odwołuje się do innej strony np. (patrz strona 109) co zupełnie nie odpowiada rzeczywistości i budzi w nas zakłopotanie. Reasumując, poradnik jest książką bardzo dobrą i zaspokoi wszystkich chcących poznać ten program.

Na marginesie sprawy. Książka jest trudna do zdobycia w księgarniach i dlatego radzę zamówić ją bezpośrednio u wydawcy lub w Studiu Atari w Szczecinie. Firmy te prowadzą sprzedaż wysyłkową.

Cena 75000 + koszty przesyłki.

SANTIAGO

AGENCJA WYDAWNICZA M&M
W-wa ul. Lanciego 12
tel. (02) 6432-509

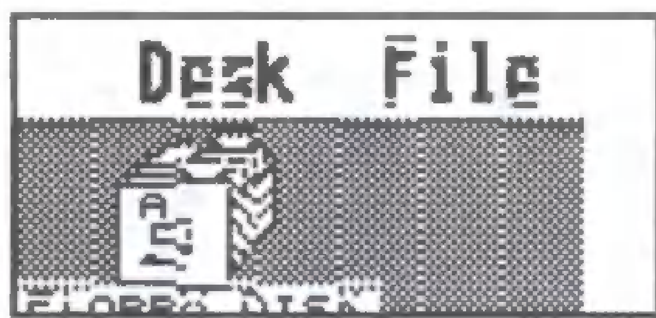
EMULATORY WYSOKIEJ ROZDZIELCZOŚCI



rys. 1



rys. 2



rys. 3



rys. 4



rys. 5

Chyba największym problemem, z jakim najczęściej spotyka się użytkownik ST jest konieczność posiadania dwóch monitorów: monochromatycznego wysokiej rozdzielczości (640 x 400) i kolorowego, dla niskiej i średniej rozdzielczości (320 x 200 i 640 x 200).

Monitor kolorowy może być zastąpiony, od biedy, telewizorem, ale trudno mówić o jakiejkolwiek pracy, a i gry wyglądają nieciekawie. Jeżeli mamy dużo pieniędzy, to możemy sobie kupić kolorowy monitor multisynchroniczny, który będzie pracował we wszystkich rozdzielczościach (najtańszy, z jakim się spotkałem, kosztował 500$). Ale co pozostaje właścicielom tylko jednego monitora?

Oczywiście możemy przywiązać go sobie do szyi, a następnie skoczyć z mostu do w miarę głębokiej rzeki. Jednak, po ostatniej letniej suszy, wiele rzek jest płytkich, a niektóre wyschły. Czyli pozostaje nam konieczność pracy (lub jak twierdzą niektórzy „męczenia się") ze specjalnym emulatorem.

Emulatory

Dla monitorów wysokiej rozdzielczości istnieją emulatory niskiej i średniej rozdzielczości, dla monitorów kolorowych (telewizorów) emulatory wysokiej rozdzielczości. Namnożyło się ich sporo, a niektóre trafiły i do naszej redakcji. Szczególnie warto się zapoznać z emulatorami wysokiej rozdzielczości i metodami ich pracy.

Szybkość emulacji możemy najczęściej ustalić, jednak tu mała uwaga. Emulator spowalnia pracę komputera. Im szybsza emulacja, tym szybciej jest tworzony obraz i niestety wolniejszy komputer.

Najprostsze...

czyli Mono Emulator v 5.00 oraz GFA Monochromkonverter v1.0 (ale długa nazwa...). Uzyskiwany obraz pozostawia sobie wiele do życzenia.

Krok naprzód

Tym krokiem jest MONULATOR v1.69. Jest to program, który pozwolił zapomnieć, o podstawowej wadzie poprzedników - o niewyraźnym obrazie. Ponadto, pozwala na powiększenie ekranu, poprzez naciśnięcie klawisza „ * ". Jakość obrazu jest zdecydowanie lepsza.

Dodatkowy program typu ACC pozwala na: wybór innego klawisza przeznaczonego do uruchomienia trybu powiększenia ekranu, zmianę szybkości emulacji i przesuwu myszy. Sam pomysł z powiększaniem obrazu jest na tyle rewelacyjny, że czasem lepiej jest pracować z MONULATOREM na kolorowym monitorze, niż na monitorze wysokiej rozdzielczości (szczególnie na 12 calowym).

Dwaj rywale

Twórca MONULATORA nie spoczął na laurach i wkrótce pojawił się kolejny emulator: MONULATOR v1.85. Ale i inni nie próżnowali - powstał chyba najlepszy (jak dotąd) z emulatorów: SeBra v1.32.

Ale zacznijmy od nowego MONULATORA. Poprawiono program ACC, który jest teraz wygodniejszy w obsłudze, dodano możliwość zmiany kolorów ekranu. Sam MONULATOR potrafi pracować z użyciem czterech różnych metod emulacji.

SeBra v1.32 to połączenie wszystkich dobrych cech swych poprzedników. Cztery różne metody, zmiana kolorów, synchronizacji ekranu (50 / 60 Hz), zmiana szybkości emulacji i przesuwu kursora myszy. I to wszystko, za pomocą odpowiedniej kombinacji klawiszy (Alternate + Control + klawisz). Jakby tego było mało, to po naciśnięciu Alternate + Control + Delete robi reset na komputerach ze starym TOS-em. Program umieszcza się w katalogu AUTO, jeżeli uruchomimy go z biurka, to okaże się, że możemy skonfigurować sobie SeBrę. Ponadto SeBra pyta się nas podczas startu systemu, czy chcemy emulacji, czy też nie.

Zamiast „ THE END'u"

Czy to już koniec rozwoju emulatorów wysokiej rozdzielczości ? Chyba nie, bo pojawiły się inne rozdzielczości, które wymagają specjalnego monitora (np. 1280 x 960 na ATARI TT) i w tym kierunku będą podążać ich twórcy...

Metody emulacji:

- metoda nieparzystych linii (rys.1)

Uzyskanie wysokiej rozdzielczości polega, na wyświetlaniu, co drugiej nieparzystej linii poziomej ekranu. W czasie emulacji komputer pracuje tak naprawdę, w średniej rozdzielczości (640 x 200).

W jaki sposób wyświetla się co drugą linię ?

W ST mamy możliwość pracy z dwoma ekranami: logicznym i fizycznym. Oba są obszarami pamięci RAM, o ściśle określonych adresach początkowych, oba też zajmują 32000 bajtów. Obraz widziany na ekranie monitora, pochodzi z obszaru pamięci przeznaczonego, dla ekranu fizycznego. Obszar ekranu logicznego jest miejscem, w którym dokonywane są wszelkie zmiany, związane z np. rysowaniem linii, przesuwaniem kursora myszy, otworzeniem nowego okna. Normalnie, adres ekranu fizycznego jest taki sam, jak adres ekranu logicznego, czyli oba ekrany pokrywają się. Dzięki temu, np. narysowana linia (obszar ekranu logiczego) jest natychmiast widoczna, na ekranie monitora.

W czasie pracy emulatora wysokiej rozdzielczości, ekrany: fizyczny i logiczny znajdują się w różnych miejscach pamięci. Zawartość pamięci ekranu fizycznego i logicznego jest inna. Rzeczywisty obraz ekranu tworzony w obszarze ekranu logicznego, przez procedury zawarte w systemie komputera, przeznaczony jest dla wysokiej rozdzielczości. Emulator dokonuje cyklicznie konwersji zawartości ekranu logicznego i efekt swojej pracy (czyli to co widzimy na monitorze), umieszcza w obszarze ekranu fizycznego.

- metoda parzystych linii (rys.2)

Identyczna, jak poprzednia, tylko odnosi się do parzystych poziomych linii ekranu.

- metoda odcieni (rys.3)

Emulator sprawdza dwa sąsiadujące ze sobą punkty w pionie, znajdujące się na ekranie logicznym. Jeżeli oba są białe, to w odpowiednim miejscu emulator umieszcza punkt w kolorze białym, jeżeli oba są czarne, to punkt będzie czarny. A co jeżeli oba są w różnych kolorach ? Wtedy punkt będzie szary. Powoduje, to że obraz jest nieco rozmyty, ale nie jest kanciasty, jak w dwóch poprzednio opisanych metodach.

Istnieje także, odmiana tej metody emulacji. Polega ona na zastąpieniu koloru szarego przez czarny (rys.4).

- metoda powiększania (rys.5)

Poprzednie metody, w mniejszym lub większym stopniu powodowały pogorszenie jakości obrazu. Tu jest odwrotnie. W danej chwili widzimy tylko pewien fragment ekranu, powiększonego w pionie dwa razy. Poruszanie się po całym ekranie jest możliwe, poprzez naciskanie wybranego klawisza (MONULATOR v1.69) lub poruszenie kursorem myszy (wtedy ekran przesuwa się w wybranym kierunku).

Proponuję wpisanie prostego programu w GFA-Basic'u, który będzie naśladował opisane poprzednio emulatory. Potrzebny będzie nam obrazek w formacie DEGASA - PI3, który zostanie poddany konwersji. Program należy uruchomić w średniej rozdzielczości.



Nie będę opisywać jego działania, gdyż wymagałoby to dokładnego przedstawienia budowy pamięci ekranu.

(pm)

Na str. 12 program emulatora

Co na dysku piszczy... (część 2)

DYSKIETKA



W pierwszej części tego cyklu przedstawiłem budowę i organizację danych na dyskietce. Uważni czytelnicy na pewno zauważyli, że pominąłem opis sektora startowego (ang. bootsector). Ten sektor jest na tyle interesujący, że wymaga osobnego artykułu.

Sektor startowy

Sektor startowy jest najważniejszym sektorem na dysku - zawiera on wszystkie informacje niezbędne do prawidłowego odczytywania i zapisywania danych. Znaczenie poszczególnych bajtów przedstawia tab. 1.

TABELA 1

bajty	nazwa pola	przeznaczenie
0-1	BRA.S	skok do procedury startowej
2-7	filler	zarezerowane
8-10	SERIAL	numer dyskietki
11-12	BPS	wielkość sektora
13	SPC	ilość sektorów przypadających na jedną zbitkę
14-15	RES	ilość zarezerwowanych sektorów
16	NFATS	ilość tablic rozmieszczenia plików (FAT)
17-18	NDIRS	ilość 32-bajtowych fragmentów w katalogu dyskietki
19-20	NSECTS	ilość sektorów na dysku
21	MEDIA	określa rodzaj dyskietki
22-23	SPF	ilość sektorów na jedną tablicę rozmieszczenia plików
24-25	SPT	ilość sektorów na ścieżce
26-27	NSIDES	ilość stron dysku
28-29	NHID	ilość schowanych sektorów

SERIAL

Podczas formatowania dyskietki w to pole wpisywana jest przypadkowa wartość. Można ją modyfikować i przez co nadać dyskowi numer porządkowy.

BPS

To pole służy do rozpoznania rozmiaru pojedynczego sektora. Może ono przyjmować następujące wartości, będące odzwierciedleniem rozmiaru sektora w bajtach:

TABELA 2
BPS
128
256
512
1024

Praktycznie nie spotyka się dysków z sektorami innymi niż 512 - to bajtowymi. Mimo, że są programy, które pozwalają formatować dyskietki z innym rozmiarem sektora, to nie należy tego robić. W przeciwnym wypadku mogą wystąpić kłopoty, włącznie z utratą danych. Dlatego, przyjmuje się, że wartość ta pozostaje stała i wynosi 512.

Pole to zapisane jest w kolejności odwrotnej.

SPC

Podaje ilość sektorów, przypadających na jedną zbitkę. Dla typowej dyskietki wynosi 2 i jest to wartość optymalna. Zmniejszenie jej do 1 pozwala na dokładniejsze wykorzystanie wolnego miejsca na dysku, za cenę spowolnienia operacji dyskowych systemu operacyjnego. Większe wartości są używane praktycznie tylko w twardych dyskach. Programy formatujące dyski (np. FastCopy PRO), pozwalają na ustawienie następujących wartości SPC: 1, 2, 4, 8.

RES

Pole to określa liczbę zarezerwowanych sektorów, włączając w to sektor startowy i dlatego, pole to ma zwykle wartość 1. Pole to zapisane jest w kolejności odwrotnej.

NFATS

W polu tym znajduje się informacja o ilości tablic rozmieszczenia plików (FAT). Typowo wartość ta wynosi 2 (dwie tablice FAT: pierwsza to oryginał, druga to kopia). Nie należy zmniejszać jej do wartości 1, gdyż w wypadku uszkodzenia jedynego FAT-a, możemy utracić bezpowrotnie dane. Często, programy formatujące dyski z 11 sektorami na ścieżce, tworzą cztery FAT-y, wtedy wartość NFATS wynosi 4. Zwiększanie ilości FAT-ów powoduje spowolnienie pracy komputera z dyskiem.

NDIRS

Pole to określa maksymalną liczbę 32-bajtowych fragmentów, jakie mogą znaleźć się w katalogu (ang. **Directory**). Powoduje to ograniczenie np. liczby plików w katalogu głównym (' A:\ ', ' B:\ '). Typowa dyskietka, może zmieścić 112 fragmentów, dlatego próba zapisania na taki dysk, np. 200-u, 1 kB-owych plików, do katalogu głównego skończy się niepowodzeniem. Po zapisaniu 112 plików, pojawi się komunikat o braku miejsca na dysku. Dobrze, że system operacyjny ST, potrafi pracować z podkatalogami (folderami). Ta sama operacja jak poprzednio, ale przy zapisie plików do podkatalogu, będzie przeprowadzona pomyślnie (podkatalog rośnie wraz ze wzrostem liczby plików i innych katalogów w nim).

Wartość pola NDIRS można określić przez podzielenie wielkości sektora (typowo 512 bajtów), przez rozmiar jednego fragmentu (32 bajty), a następnie pomnożeniu przez liczbę sektorów zajmowanych przez katalog dyskietki (typowo 7).

Przeliczenie w drugą stroną, z wartości NDIRS, do liczby sektorów zajmowanych przez katalog dyskietki, jest równie proste i nie powinno stanowić większego problemu.

Pole NDIRS zapisane jest w kolejności odwrotnej.

NSECTS

Pole to podaje ilość sektorów znajdujących się na dysku. Liczbę ścieżek na dysku otrzymujemy z tej wartości, dzieląc ją przez liczbę sektorów na ścieżce (SPT) i liczbę stron dysku (NSIDES). NSECTS uwzględnia wszystkie sektory, także zarezerwowane.

Pole to zapisywane jest w kolejności odwrotnej.

MEDIA

Określa rodzaj dysku. System operacyjny ST nie używa tego pola, może być ono wykorzystywane przez inne systemy operacyjne, dlatego nie należy ingerować w to pole.

SPF

Podaje ilość sektorów zajmowanych przez jeden FAT. Pole to nie może mieć wartości dowolnej. Aby tą wartość uzyskać, należy podzielić liczbę sektorów (SECTS) przez ilość sektorów, przypadających na jedną zbitkę (SPC). Następnie mnożymy przez 1.5 (12-o bitowe pozycje FAT - dla dyskietek) lub przez 2 (16-o bitowe pozycje FAT - dla twardych dysków). Po podzieleniu przez rozmiar sektora (BPS), czyli przez 512,

otrzymamy szukaną liczbę sektorów. Nie będzie to liczba całkowita (np. 2.109) i dlatego należy ją zaokrąglić w górę dla 2.109 będzie to 3).

Łatwo policzyć, że coś dziwnie dobierali wartość SPF, programiści piszący system operacyjny ST. Zamiast spodziewanej liczby 3, dla dyskietki dwustronnej (DS) i liczby 2, dla dyskietki jednostronnej (SS), dyskietka jest formatowana z biurka, z wartością SPF równą 5.

Miało to zapewne na celu, takie przesunięcie obszaru danych, aby zaczynał się on od nowej ścieżki. Najprawdopodobniej, chciano zabezpieczyć sektory należące do katalogu dysku, przed przypadkowym uszkodzeniem, jakie mogło by wystąpić podczas zapisu danych do jednego z pierwszych sektorów obszaru danych (czyli, sektora znajdującego się na tej samej ścieżce, co katalog).

Pole to jest zapisywane w kolejności odwrotnej.

SPT

Pole to określa liczbę sektorów przypadających na jedną ścieżkę. Dla dyskietki DD (takich jakie normalnie używa się z ST) wynosić ono może od 1 do 11. Wartość ta jest ustalana przez użytkownika, przez podanie liczby sektorów w programie formatującym dyskietki.

Nie ma sensu formatować dysk z liczbą sektorów na ścieżkę mniejszą niż 9. Dyskietki formatowane na 9 sektorów są „pewne" w pracy i nie ma z nimi problemów (chyba, że ktoś porysuje ją np. gwoździem lub będzie ją używał jako podstawki pod gorące naczynia).

Wielu użytkowników formatuje dyski na 10 sektorów. Z takimi dyskami raczej nie ma kłopotów. W specjalnych przypadkach można formatować na 11 sektorów, ale odradzam, gdyż z własnego doświadczenia wiem, że niektóre programy są wybredne i z takimi dyskami „nie chodzą", utrata danych też nie należy do rzadkości.

Pole to jest zapisywane w kolejności odwrotnej.

NSIDES

Wartość tego pola podaje nam liczbę stron dysku. Dla dyskietki jednostronnej wynosi 1, dla dwustronnej 2. Pole to jest wykorzystywane przez ST i umożliwia identyfikację dyskietki, dzięki czemu stacja dwustronna może pracować z dyskietką zapisaną tylko na jednej stronie. Próba odwrotna (praca z dyskietką zapisaną na obu stronach, przy stacji jednostronnej), jest skazana na niepowodzenie.

Pole to jest zapisywane w kolejności odwrotnej.

NHID

Pole to określa liczbę „ schowanych sektorów ". Pierwsze wersje systemu operacyjnego ST ignorowały to pole (zawsze wartość 0). Czy późniejsze używają NHID nie wiem.

Pole to zapisywane jest w kolejności odwrotnej.

KONIEC ?

Mogłoby się wydawać, że to już koniec opisu dyskietki. Wszystko, co miało być opisane, zostało opisane, ale nie. Można się o tym łatwo przekonać.

Proponuję dokonać małego eksperymentu, wystartuj drogi Czytelnku, komputer z dowolnej dyskietki, ale takiej aby ukazało się biurko. Następnie weź dyskietkę z grą lub programem demonstracyjnym i spróbuj obejrzeć czy są jakieś pliki.

Jeżeli masz szczęście (?!), to pojawi się komunikat o znalezionym błędzie, kiedyindziej komputer się zresetuje albo zawiesi. Może też zobaczysz ikony podpisane jakimiś bezładnymi tekstami.

Teraz zresetuj komputer. Jeżeli np. gra załaduje się sama, to znaczy, że stało się coś, co nie powinno się stać, bo jak gra mogła sama z siebie się wczytać i uruchomić ?.

Wyjaśnienie tego „cuda" jest proste - programiści, którzy napisali system operacyjny ST, w dużej mierze wzorowali się na MS - DOS (systemie komputera IBM PC). Stąd zaczerpnęli pomysł wykorzystania pozostałych, nieużywanych bajtów w sektorze startowym. W tym obszarze, można umieścić krótki program ładujący np. grę.

Bliższe informacje, o tym jak tego dokonać, znajdą się w kolejnym odcinku cyklu „Co na dysku piszczy... ".

Dlaczego stosuje się podział ścieżek na sektory ?

1. Na uszkodzonej ścieżce można zapisać dane, ale w czasie ich odczytu okazałoby się, że są one zniekształcone, czyli cała ścieżka musiałaby pozostać niewykorzystana. Przy podziale na sektory, część z nich nadawałaby się do użytku, a część nie (w miejscu uszkodzenia). Dzięki temu nie tracimy miejsca na dysku.
2. Załóżmy, że na jednej ścieżce można zapisać 5 kB danych. Chemy zapisać plik o długości 1 kB, a ścieżka nie jest podzielona na sektory. Ile zajmie ten ten plik miejsca na dysku ? Okazuje się, że 1 kB, 4 kB pozostanie wolne i nie zostaną tam umieszczone inne pliki. Kilkadziesiąt taki plików mogłoby „zapchać" cały dysk, mimo, że w sumie zajmowałyby dziesięciokrotnie mniej miejsca na dysku. Podzielmy więc ścieżki, każdą na dziesięć 0.5 kB-owych fragmentów (sektorów). Zapisany plik o długości 1 kB zajmuje tylko dwa sektory, pozostałe osiem sektorów może być wykorzystane na zapis innych plików.

(pm)



DIALOG Z KOMPUTEREM
GEM

Artykuł ten otwiera nowy cykl, poświęcony tym użytkownikom, którzy zajmują się programowaniem ATARI ST. Nie będzie to opis stosunków, jakie panują między komputerem, a jego użytkownikiem, ale sposób, w jaki można komunikować się ze swoim ST. Dla tych, którzy nie wiedzą jeszcze, jak wykorzystać „ROM" swojego komputera, do tego, aby ich programy stały się bardziej estetyczne, przejrzyste i co najważniejsze, aby komunikacja z użytkownikiem była co najmniej dostateczna, cykl ten będzie odpowiednim materiałem.

Jak każdemu wiadomo, ROM ATARI ST to nic innego, jak system operacyjny TOS (Tramiel Operating System). Podzielony jest on na trzy bloki (rys.1):

rys. 1

GEM - system operacyjny odpowiedzialny za wszelkiego rodzaju operacje graficzne. Składa się on z trzech zasadniczych części:

VDI - zawiera procedury wykonujące elementarne operacje graficzne np.: rysowanie punktu, odcinka, okręgu itp.

AES - zawiera procedury zarządzające ekranem i akcesoriami, oraz bibliotekę podprogramów obsługujących menu, okna, mysz itp.

GEMDOS - zarządzanie pamięcią, operacje na plikach, obsługa drukarki, monitora, interfejsu szeregowego, portu MIDI, klawiatury.

XBIOS - zapewnia obsługę funkcji, związanych z architekturą sprzętową rodziny ST np. obsługa układu dzwiękowego.

Funkcje dodatkowe - zestaw procedur graficznych, wspomagających GEM.

Ja chciałbym zająć się tylko częścią TOS'u, która odpowiedzialna jest za operacje graficzne. Napewno wielu z Was, nie zdaje sobie sprawy, jak wielkie możliwości daje programiście właśnie GEM. Ale nie będę opisywał wszystkiego. Celem tego cyklu jest pokazanie, w jaki sposób za pomocą „komputera", można obsługiwać okna (rys.2), okna dialogowe (rys.3), tzw. skrzynki alarmowe (rys.4), menu typu drop-down (rys.5) oraz ikony (rys.6).

rys. 2

Umiejętność posługiwania się wyżej wymienionymi „obiektami", pozwoli Państwu na wykorzystanie „możliwości graficznych", drzemiących w ROM'ie tego komputera, co w konsekwencji wpłynie na wygląd i niezawodność tworzonych programów, a po drodze poznacie Państwo „kilka" zagadnień, związanych z ATARI ST.

Wewnątrz AES'u umieszczona jest biblioteka procedur, obsługujących wyżej wymienione obiekty (okna, ikony itp.). Może to być np. przemieszczanie obiektów graficznych, odczytywanie pozycji kursora, myszy, wyświetlanie ikon, obsługa poleceń wprowadzanych przez użytkownika, wybór opcji za pomocą menu, tworzenie okien, manipulacja oknami itp. Jak widać możliwości te są naprawdę duże.

rys. 3

Dzisiaj chciałbym zacząć od okien dialogowych. Przypominają one ankietę, w której zamiast na papierze, odpowiedzi wpisuje się na ekranie. Zakres zastosowań okien dialogowych nie jest niczym ograniczony. Można ich także używać do budowy własnych menu, ale o tym później. W celu uproszczenia programowania okien dialogowych, AES przyjmuje na siebie następujące obowiązki:

rys. 4

rys. 5

Wyświetlanie okien dialogowych

Całkowity nadzór nad wszystkimi działaniami użytkownika tj. wprowadzanie tekstów, selekcja pól itd.

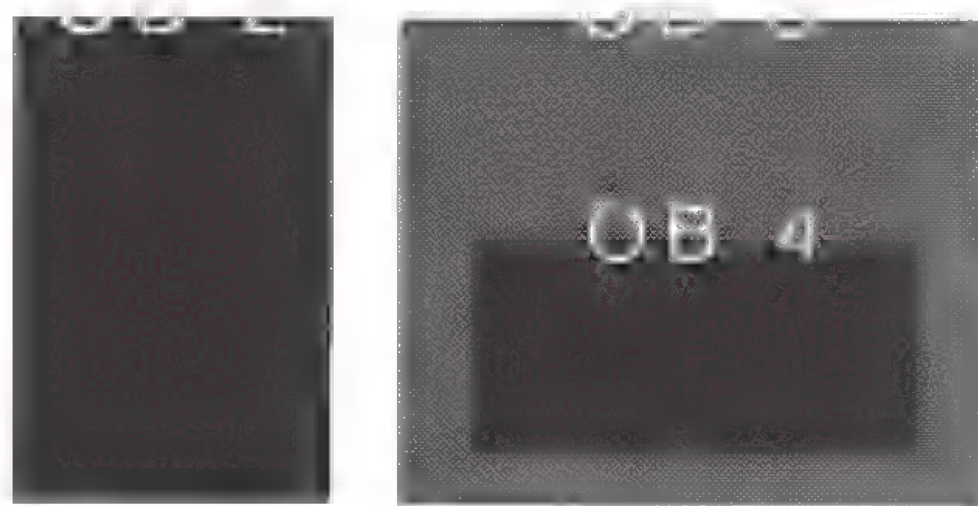
Okno dialogowe nie jest związane z obramowaniem okna na ekranie, tzn. że może ono nachodzić na jedno lub kilka okien. W chwili ukończenia „dialogu" z oknem dialogowym, AES określa, które przedmioty zostały zmienione. Przykład takiego okna przedstawia rys. 3. Wszystkie elementy znajdujące się wewnątrz okna dialogowego (teksty, prostokąty, rysunki itp.), nazywać będziemy przedmiotami. Uwaga - prostokąt będący „obramowaniem" okna dialogowego jest też przedmiotem.

rys. 6

Programista ma możliwość tworzenia własnych przedmiotów np. ikon własnego projektu. Proszę spojrzeć na rys. 7. Przedstawia on nieco skromne okno dialogowe, które nie reprezentuje sobą praktycznie nic, ale pozwoli nam zrozumieć pewne zagadnienia.

Na ekranie znajduje się duży prostokąt (ob1), a wewnątrz dwa mniejsze (ob2 i ob3). Jeden z tych „mniejszych" (ob3) zawiera jeszcze mniejszy (ob4). Między tymi przedmiotami zachodzą pewne relacje. Można to określić jako strukturę drzewa. OB1 zawiera bezpośrednio pod sobą OB2 i OB3 oraz pośrednio OB4, przedmiot OB3 zawiera bezpośrednio element OB4. Strukturę powiązań między tymi obiektami przedstawia kolejny rysunek 8.

Jak wygląda rozkład przedmiotów w pamięci ? Otóż do tego celu potrzebne będą wskaźniki (adresy). Informacje o obiektach mogą być przechowywane w pamięci w dowolnym porządku. Każdemu utworzonemu przedmiotowi przydzielone są 24 bajty (12 słów, 1 słowo = 2 Bajty = 16 bitów). Sześć z nich zarezerwowane są właśnie na wskaźniki. Każdemu przedmiotowi przydzielona jest liczba (numer porządkowy)



rys. 7

od 0 (pierwszy element) do n-1 (ostatni element, gdzie n - liczba przedmiotów). Jeden ze wskaźników zawiera numer przedmiotu, który jest mu przyporządkowany. (w tym przypadku jeden z trzech wskaźników przedmiotu OB3, zawiera numer przedmiotu OB4). Jeżeli obiekt nie posiada żadnego podporządkowanego sobie przedmiotu (np. OB2), to wskaźnik ten zawierać będzie wartość -1 ($FFFF).

Ale co zrobić jak jeden przedmiot ma pod sobą bezpośrednio 10 elementów. Przecież nie może wskazywać on na każdy element po kolei, gdyż musiałby posiadać aż 10 wskaźników. Do tego celu wykorzystywany jest trzeci wskaźnik. Przykładowo przedmiot o indeksie 1 zawiera bespośrednio pod sobą 4 elementy o indeksach 2, 3, 4, 5. Trzeci wskaźnik elementu o indeksie 2 wskazuje na element 3, ten z kolei wskazuje na element 4, a 4 na 5. Element o indeksie 5 jest ostatnim, nie ma już na co wskazywać, więc wskazuje z powrotem na obiekt o wyższym poziomie, czyli 1.

Jak już wcześniej wspomniałem, na każdy przedmiot przypada 24- ro bajtowy zapis. Zorganizowany jest on w następujący sposób.

Słowo 0 - indeks następnego przedmiotu z grupy przedmiotów podporządkowanych.
Słowo 1 - indeks pierwszego przedmiotu podporządkowanego.
Słowo 2 - indeks ostatniego przedmiotu podporządkowanego.
Słowo 3 - typ przedmiotu
Słowo 4 - wskaźniki przedmiotowe. Określają jaką funkcję powinien spełniać przedmiot w oknie dialogowym.
Słowo 5 - status przedmiotu. Określa w jaki sposób przedmiot powinien być wyświetlony na ekranie.
Słowo 6 i 7 - określenie własności przedmiotu.
Słowo 8 - X-owa współrzędna przedmiotu.
Słowo 9 - Y-owa współrzędna przedmiotu.
Słowo 10 - szerokość przedmiotu.
Słowo 11 - wysokość przedmiotu.

W oknie dialogowym, jak zdążyli Państwo nieco wcześniej zauważyć, można umieścić przedmioty o różnych typach. Jakie to mogą być przedmioty, jakie jest ich przeznaczenie?

Każdy typ przedmiotów, posiada swoją liczbę, która umieszczona jest w słowie 3 obszaru pamięci, jakim jest określony przedmiot (owe 24 bajty). Poniżej znajduje się charakterystyka wszystkich typów przedmiotów, ich numer oraz nazwa (obowiązująca w programach do tworzenia m.in. okien dialogowych np. GEM RESOURSE CONSTRUCTION SET, INTERFACE czy inne).

20 **BOX** - prostokąt
21 **TEXT** - tekst graficzny; słowa 6 i 7 stanowią adres struktury TED INFO
22 **BOXTEXT** - prostokąt zawierający tekst; słowa 6 i 7 stanowią adres struktury TED INFO
23 **IMAGE** - rysunek rastrowy; słowa 6 i 7 stanowią wskaźnik struktury BITBLK
24 **PROGDEF** - przedmiot definiowany przez programistę; słowa 6 i 7 stanowią adres struktury APPLBLK
25 **IBOX** - niewidoczny prostokąt
26 **BUTTON** - scentrowany tekst graficzny w prostokącie; słowa 6 i 7 zawierają adres tekstu przeznaczonego do wyprowadzenia. Tekst musi być zakończony bajtem zerowym
27 **BOXCHAR** - prostokąt zawierający tylko jedną, ustawioną w środku literę. Bardziej znaczący bajt słowa 6, zawiera kod znaku
28 **STRING** - tekst graficzny; słowa 6 i 7 zawierają wskaźnik tekstu przeznaczonego do wyprowadzenia
29 **FTEXT** - sformatowany tekst graficzny; słowa 6 i 7 zawierają adres struktury TED INFO
30 **FBOXTEXT** - prostokąt zawierający sformatowany tekst graficzny; słowa 6 i 7 zawierają wskaźnik struktury TED INFO
31 **ICON** - ikona; słowa 6 i 7 zawierają wskaźnik struktury ICONBLK
32 **TITLE** - tekst graficzny; słowa 6 i 7 zawierają adres tekstu przeznaczonego do wyprowadzenia, który musi być zakończony bajtem zerowym.

Na temat typów przedmiotów, było by na tyle, chociaż występuje tutaj parę niuansów. Co to są struktury TED INFO, BITBLK, APPLBLK, ICONBLK ? Spróbuję to Państwu jeszcze w miarę dokładnie wyjaśnić.

Struktura to nic innego, jak pewien obszar pamięci, zawierający parametry potrzebne przy wykonywaniu określonych czynności.

TED INFO - służy ona do wyprowadzania sformatowanego tekstu graficznego. Wykorzystywana jest ona przez przedmioty TEXT, BOXTEXT, FTEXT, FBOXTEXT, których 6 i 7 słowo jest adresem właśnie tej struktury. W pamięci zorganizowana ona jest w następujący sposób:

Słowo 0 i 1 - (ptext) adres wyprowadzanego ciągu znaków, który musi być zakończony bajtem zerowym
Słowo 2 i 3 - (ptmplt) adres ciągu, który ma zostać zmiksowany z ptext. Jest on wykorzystywany jako maska wejściowa dla wprowadzanego tekstu. Pozycja wprowadzanego przez użytkownika znaku, musi być wypełniona "_".
Słowo 4 i 5 - (pvalid) adres ciągu określającego, które typy znaków są znakami dozwolonymi przy wprowadzaniu tekstu do przedmiotu. Każda wprowadzana pozycja, może zostać określona oddzielnie.
9 - dozwolone są cyfry od 0 - 9
A - tylko duże litery A - Z oraz spacje
a - tylko duże i małe litery oraz spacje
N - tylko cyfry 0 - 9, duże litery A - Z i spacje
X - wszystkie znaki na jakie zezwala TOS
Słowo 6 - (font) Numer zbioru znaków, który powinien być wykorzystywany w trakcie wyprowadzania informacji na ekran (3 - zwykły zbiór znaków, 5 - zredukowany zbiór znaków)
Słowo 7 - (resvdl) rezerwowany dla przyszłych zastosowań
Słowo 8 - (just) sposób formatowania tekstu (0 - justowany lewostronnie, 1 - justowany prawostronnie, 2 - centrowany)
Słowo 9 - (color) kolor
Słowo 10 - (resvd2) rezerwowany dla przyszłych zastosowań
Słowo 11 - (thickness) grubość prostokątnego obramowania (0 - bez obramowania, 1-128 grubość obramowania, kierunek do wewnątz, (-1)-(-127) grubość obramowania, kierunek na zewnątrz)
Słowo 12 - (txlen) długość ciągu znaków ptext
Słowo 13 - (tpplen) długość ciągu znaków ptmplt

BITBLK - struktura zawierająca uzupełniające informacje o rozmiarze i wyglądzie obiektu graficznego. W pamięci zorganizowana ona jest w następujący sposób:
Słowo 0 i 1 - (pdata) wskaźnik tablicy bitów z jakich składa się obiekt graficzny typu IMAGE (rysunek)
Słowo 2 - (bi_wp) szerokość rysowanego obiektu w bajtach
Słowo 3 - (bi_hi) wysokość obiektu w pikslach
Słowo 4 - (bi_x) X-owa wspł. punktów w tablicy
Słowo 5 - (bi_y) pozycja pierwszego rzędu w tablicy
Słowo 6 - (bi_color) kolor obiektu graficznego

APPLBLK - za pomocą przedmiotu typu PROGDEF, w strukturze drzewa (rys 8), na miejsce przedmiotu można umieścić

swój własny program. Słowa 6 i 7 przedmiotu typu PRODEF zawierają adres struktury APPLBLK. Zorganizowana ona jest w pamięci w następujący sposób:

Słowo 0 i 1 – (ab_code) adres programu

Słowo 2 i 3 – (ab_parm) wartość opcjonalna dla AES przekazywana do prorgramu

ICONBLK – struktura zawierająca szczegółowe informacje, dotyczące reprezentacji ikon. Ikony są reprezentowane przez obiekt typu ICON. Słowa 6 i 7 określające przedmiot właśnie tego typu, zawierają adres struktury ICONBLK. Zorganizowana jest ona w pamięci w następujący sposób:

Słowo 0 i 1 – (ib_pmask) adres tablicy słów określających szablon ikony

Słowo 2 i 3 – (ib_pdata) adres tablicy słów opisujących wygląd ikony.

Słowo 4 i 5 – (ib_ptext) adres wskazujący na ciąg znaków mający pojawić się w ikonie. Ostatnim znakiem ciągu musi być bajt zerowy

Słowo 6 – (ib_char) kod litery mający pojawić się w ikonie

Słowo 7 – (ib_xchar) X-owa wspł. litery ib_char

Słowo 8 – (ib_ychar) Y-owa wspł. litery ib_char

Słowo 9 – (ib_xicon) X-owa wspł. ikony

Słowo 10 – (ib_yicon) Y-owa wspł. ikony

Słowo 11 – (ib_wicon) szerokość ikony w pikslach (wartość ta musi być podzielna przez 16)

Słowo 12 – (ib_hicon) wysokość ikony w pikslach

Słowo 13 – (ib_xtext) X-owa wspł. tekstu ikony

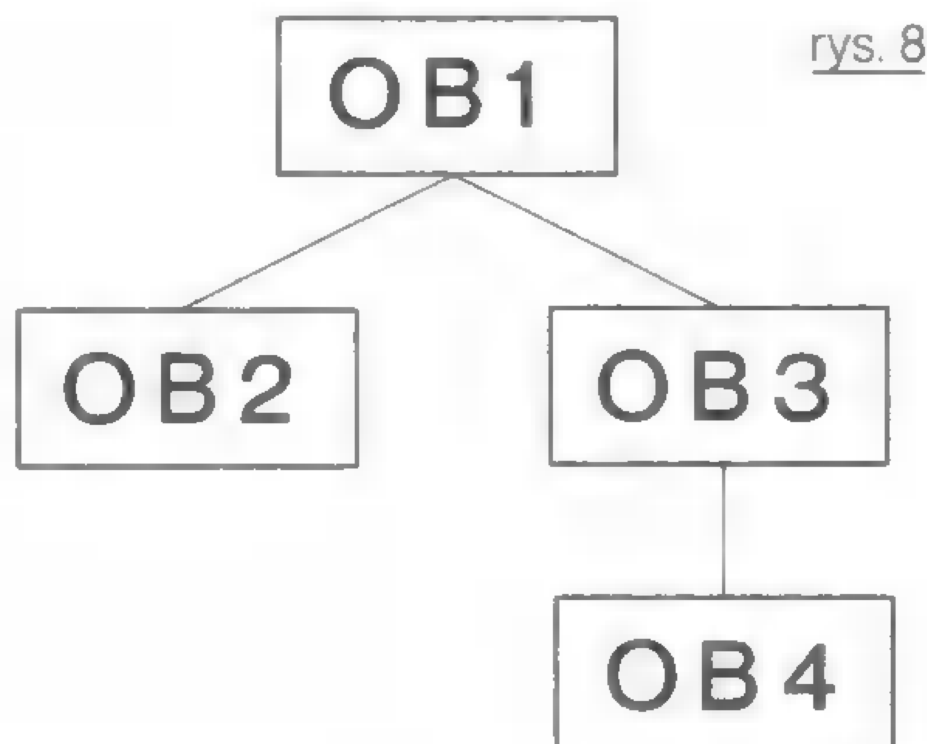
Słowo 14 – (ib_ytext) Y-owa wspł. tekstu ikony

Słowo 15 – (ib_wtext) szerokość tekstu ikony w pikslach

Słowo 16 – (ib_htext) wysokość tekstu ikony w pikslach

Typy przedmiotów, oraz rodzaje struktur mamy już prawie za sobą. Pozostaje już tylko kwestia „przetrawienia" tego całego materiału. Nie jest to wcale materiał prosty do opanowania (ja czytałem go 3 razy i dopiero za 4 razem, zaczeło rozjaśniać mi się w głowie), ale opanowanie go, pozwoli na zrozumienie wielu ciekawych, ale niestety trudnych rzeczy. Na koniec pozwolą Państwo, że dodam jeszcze odrobinę materiału, aby temat ten został zakończony.

6 i 7 słowo przedmiotów typu BOX, IBOX, i BOXCHAR nie wskazują na żadną strukturę. Dlaczego ? Do reprezentacji tego typu przedmiotów nie potrzeba zbyt wiele danych, mało tego – dane te mieszczą się w dwóch słowach, a zatem mogą się one zawierać w 6 i 7 słowie reprezentowanego przedmiotu. Słowo 7, jest kolorem przedmiotu. Najbardziej znaczący bajt słowa 6 jest kodem literowym dla BOXCHAR. Najmniej znaczący bajt słowa 6, określa szerokość obramowania danego przedmiotu (0 – bez obramowania, 1– 128 grubość obramowania, kierunek do wewnątz, (-1)-(-127) grubość obramowania, kierunek na zewnątrz).



rys. 8

Niejasne może być jeszcze pojęcie miksowania tekstów (struktura TED INFO). Aby to zrozumieć, posłużmy się przykładem:

Niech **ptext** będzie tekstem dla określenia daty (mies. dzień rok), która w czasie wyświetlania powinna pojawić się jako wartość podstawowa (wejściowa) :"280874" (28.08.74)

Niech **ptmplt** będzie szablonem określającym linie wprowadzającą "Data urodzenia:__/__/__"

Pvalid określa, że jako dane wejściowe dozwolone są tylko cyfry. Pvalid zawiera więc ciąg "999999"

Pierwsze dwa ciągi w trakcie wyprowadzania, zostają nałożone na siebie (zmiksowane). Jako wynik otrzymujemy ciąg "Data urodzenia:28/08/74"

W przypadku, gdy użytkownik wprowadzi cyfry "010191", pojawi się tam napis "Data urodzenia: 01/01/91"

W tym numerze to już naprawdę wszystko. Wiedzę, którą Państwo posiedli (mam nadzieję), bądź posiądą po kilkakrotnym przeczytaniu tego artykułu, wykorzystamy w następnym numerze, w którym powiemy sobie w jaki sposób sterować tymi oknami np. w GFA BASICU.

Codie

Literatura:
[1] GEM ST podręcznik programisty
[2] GFA BASIC - Stefan Nawrocki


```
'
' EMULACJA WYSOKIEJ ROZDZIELCZOŚCI
' Autor: Przemysław Mazurek
' ST-Fan 2/93
' Wszelkie prawa zastrzeżone
'
obrazek__buf$=SPACE$(32034)
obrazek__buf%=VARPTR(obrazek__buf$)
'
PRINT "Wybierz plik"
'
FILESELECT "\*.pi3",plik$,plik$
OPEN "I",#1,plik$
BGET #1,obrazek__buf%,32034
CLOSE
'
DO
 obrazek%=obrazek__buf%+34
 ekran%=XBIOS(2)
 '
 CLS
 PRINT " Wybierz metodę emulacji:"
 PRINT " 1 - nieparzyste linie"
 PRINT " 2 - parzyste linie"
 PRINT " 3 - odcienie"
 PRINT " 4 - bez szarości"
 PRINT " 5 - powiększanie"
 '
 INPUT pyt%
 CLS
 '
 IF pyt%=1
 @nieparzyste
 ENDIF
 IF pyt%=2
 @parzyste
 ENDIF
 IF pyt%=3
 @kolory2
 @odcienie
 ENDIF
 IF pyt%=4
 @kolory1
 @odcienie
 ENDIF
 IF pyt%=5
 @powiekszanie
 ENDIF
 '
 DO
 EXIT IF INKEY$<>""
 LOOP
LOOP
'
PROCEDURE kolory1
 SETCOLOR 0,7,7,7
 SETCOLOR 3,0,0,0
 SETCOLOR 1,0,0,0
 SETCOLOR 2,0,0,0
RETURN
'
PROCEDURE kolory2
 SETCOLOR 0,7,7,7
 SETCOLOR 3,0,0,0
 SETCOLOR 1,3,3,3
 SETCOLOR 2,3,3,3
RETURN
'
'
PROCEDURE nieparzyste
 @kolory1
 FOR i%=0 TO 199
 obrazek%=obrazek%+80
 FOR j%=0 TO 39
 SDPOKE ekran%,DPEEK(obrazek%)
 obrazek%=obrazek%+2
 ekran%=ekran%+4
 NEXT j%
 NEXT i%
RETURN
'
PROCEDURE parzyste
 @kolory1
 FOR i%=0 TO 199
 FOR j%=0 TO 39
 SDPOKE ekran%,DPEEK(obrazek%)
 obrazek%=obrazek%+2
 ekran%=ekran%+4
 NEXT j%
 obrazek%=obrazek%+80
 NEXT i%
RETURN
'
PROCEDURE odcienie
 FOR i%=0 TO 199
 FOR j%=0 TO 39
 SDPOKE ekran%,DPEEK(obrazek%)
 SDPOKE ekran%+2,DPEEK(obrazek%+80)
 obrazek%=obrazek%+2
 ekran%=ekran%+4
 NEXT j%
 obrazek%=obrazek%+80
 NEXT i%
RETURN
'
PROCEDURE powiekszanie
 @kolory1
 FOR i%=0 TO 199
 FOR j%=0 TO 40
 SDPOKE ekran%,DPEEK(obrazek%)
 obrazek%=obrazek%+2
 ekran%=ekran%+4
 NEXT j%
 NEXT i%
RETURN
```

PROGRAMOWANIE NA ATARI ST
GFA BASIC
LEKCJA 5

Nie chciałbym, co niektórych zmartwić, ale wiedzę, którą Państwo posiedli, analizując pierwsze trzy lekcje naszego kursu programowania w GFA BASIC'u, to dopiero wstęp. Temat obejmujący rodzaje zmiennych, oraz podstawowe pojęcia, jest obszerny i żmudny, ale bez niego nie można mówić o programowaniu. Oczywiście, mogłem zmieścić go w jednej, bądź dwóch lekcjach, ale nie sądzę, by po ich przeczytaniu, część użytkowników (przynajmniej tych początkujących), zrozumiała w stopniu dostatecznym opisywane zagadnienia.
Teraz, kiedy tą żmudną teorię mamy już za sobą, czeka Nas dużo, ale za to ciekawej pracy. Zaznaczam z góry, że nie będę wracał do poprzednich tematów. W związku z tym, przerobiony materiał, należałoby mieć przynajmniej w „główce". Ideałem byłoby opanowanie materiału do tego stopnia, aby znalazł się on w „paluszku", ale to już nie jest warunek konieczny.

W dzisiejszej lekcji chciałbym zapoznać Państwa, z podstawowymi i niemalże najważniejszymi instrukcjami, które są "szkieletem" wszystkich języków wyższego poziomu (np. Basic, Pascal, C, itp.). Należą do nich: instrukcje iteracyjne, instrukcja skoku oraz instrukcja warunkowa. Program, a właściwie polecenia, z których się składa, wykonywane są kolejno, jedno po drugim, zgodnie z ich ułożeniem. Ale nie ma reguły bez wyjątków. Instrukcje, które dzisiaj poznamy, mogą ten porządek zakłócać. Przekonają się Państwo o tym, po dokładnym przeanalizowaniu dzisiejszej lekcji.

Instrukcja skoku - GOTO

Wykorzystywana jest do swobodnego poruszania się po programie (przeskakiwania z jednego miejsca w drugie), chociaż nie jest zalecana przez programistów. Powoduje ona przekazanie sterowania do instrukcji, poprzedzonej etykietą, której nazwa znajduje się w komendzie GOTO, jako jej jedyny parametr np.:

GOTO nazwa etykiety

Zastosowanie i działanie tej instrukcji, przedstawia następujący program: (strzałki pokazują kolejność wykonywania poleceń)

```
Print "linia 1"
Print "linia 2"
Goto Dalej
Print "linia 3"
Print "linia 4"
Dalej:
Print "linia 5"
Print "linia 6"
End
```

Jak Państwo zauważyli, polecenia: PRINT "LINIA 3" i PRINT "LINIA 4" nie zostały wykonane, gdyż na ekranie pojawiłyby się kolejno komunikaty - LINIA 3" i "LINIA 4". Wykonane zostały pierwsze dwa polecenia (PRINT "LINIA 1" i PRINT "LINIA 2"). "GOTO Dalej" spowodowało przeskok do instrukcji, znajdującej się tuż pod etykietą "Dalej" (PRINT "LINIA 5") i dalsze kontynuowanie programu.

Istnieje jeszcze jedna instrukcja skoku - GOSUB, ale jej omówienie odłóżmy na inny termin, gdyż wymaga to znajomości procedur.

Instrukcja warunkowa IF THEN ELSE

Ma ona za zadanie, wykonanie pewnej czynności, w zależności od tego czy podany w niej warunek jest spełniony, czy też nie. Istotę tej instrukcji bardzo dobrze oddaje jej "polskie znaczenie". IF ... THEN ... ELSE ... - JEŚLI (warunek jest spełniony) WTEDY (wykonaj ciąg instrukcji) W PRZECIWNYM RAZIE (gdy warunek nie jest spełniony, wykonaj inny ciąg instrukcji). Warunki, jakie mogą być umieszczone w instrukcji, są typu: A>2, B>=3, x%<>0, a$="A" itp. Dla głębszego zrozumienia sensu stosowania tej instrukcji, posłużmy się przykładem:

```
Input "Wprowadź liczbę ";X
If X>0 Then
Print "Dodatnia"
Else
Print "Niedodatnia"
Endif
```

Jako parametr, posłuży nam liczba wprowadzona z klawiatury przez użytkownika (Proszę wprowadzić raz liczbę dodatnią, a raz ujemną - ze znakiem "-"). Program rozpoznaje, czy wprowadzona liczba jest dodatnia (większa od 0), czy też nie. I w zależności od tego wyświetla odpowiedni komunikat: "Dodatnia", bądź "Niedodatnia". Jak należy poprawnie przeczytać taki program ?
JEŚLI liczba zawarta w zmiennej X jest większa od 0 (dodatnia) WTEDY napisz "Dodatnia", W PRZECIWNYM WYPADKU (jeśli nie jest większa od 0) napisz "Niedodatnia".
Słowo ENDIF jest tzw. kluczem, który zamyka instrukcję warunkową. UWAGA: Nie można go pominąć !!! Instrukcja ta może natomiast występować bez słowa ELSE np.:

```
If X>0 Then
Print "Dodatnia"
Endif
```

W GFA BASICU warunki występują nie tylko w instrukcjach tego typu. Istnieje jeszcze kilka "poleceń", w których odgrywa on znaczną rolę. Warunek może składać się z kilku członów, połączonych operatorami AND (i) oraz OR (lub). A oto przykład takiego "złożonego warunku":

A>0 AND B<2 AND X<>0

Na całość tego warunku wpływają aż trzy człony. Żeby został on spełniony, jednocześnie muszą być spełnione wszystkie trzy relacje (A>0 i B<2 i X<>0). Jeżeli, przynajmniej jedna z nich nie będzie spełniona, cały warunek nie będzie uznany za prawdziwy. Spróbujmy to teraz zastosować w instrukcji, którą poznaliśmy.

```
Input "Podaj liczbe A i B";A,B
If A=2 AND B=4 Then
Print "Brawo !!!"
Endif
```

JEŚLI A będzie równe 2 "i" B równe 4 WTEDY napisz "Brawo". Proszę zwrócić uwagę, aby komunikat "Brawo" został wyświetlony, warunek "A=2 AND B=4" musi być prawdziwy. Natomiast, aby warunek ten był prawdziwy, składające się na niego relacje (A=2 i B=4), muszą być jednocześnie spełnione. Jeżeli zaistnieje sytuacja, że jedna relacja jest prawdziwa, a druga nie, to automatycznie warunek nie będzie spełniony.
Operator OR jest na tyle elastyczny, że pozwala na "pozytywne wykonanie operacji" w przypadku gdy prawdziwy będzie przynajmniej jeden człon postawionego warunku. Spróbujmy przedstawić tą sytuacje na tym samym przykładzie. W powyższym przykładzie wymieńmy operator AND na OR. Komunikat "Brawo" zostanie wyświetlony dopiero wtedy, gdy **przynajmniej** jedna z relacji składających się na całość WARUNKU, zostanie spełniona (A=2 **lub** B=4).
Instrukcja warunkowa wykorzystywana jest przede wszystkim, do kontrolowania pracy programu, w zależności od parametrów, np. w zależności od wybranej funkcji, jakimi dysponuje program, wykonywany jest odpowiedni podprogram, czy procedura obsługująca daną funkcję.

Instrukcje iteracyjne

Służą do organizowania cykli programowych tj. wielokrotnego wykonywania pewnych sekwencji instrukcji. W GFA BASICU występują ich cztery rodzaje:

FOR....NEXT
REPEAT....UNTIL
DO....LOOP
WHILE....WEND

Instrukcje iteracyjną FOR...NEXT, stosuje się w celu wykonywania pewnej grupy instrukcji, w przypadku gdy liczba powtórzeń jest znana w danym miejscu programu. A oto przykład:

```
For X=1 To 10
Print "GFA BASIC"
Next X
```

Strzałki pokazują kolejność wykonywania instrukcji. Efektem działania programu jest wyświetlenie na ekranie komputera, 10 razy "GFA BASIC". Dlaczego 10 ? W pierwszej lini pętli FOR X=1 to 10, następuje przypisanie licznikowi, którym będzie zmienna liczbowa X, wartości początkowej 1. Liczba umieszczona po słowie "To" jest wartością końcową licznika, tzn. że jeżeli osiągnie on wartość 10, pętla zostanie opuszczona. Podstawową zasadą do zrozumienia istoty pętli FOR...NEXT jest fakt, że po każdorazowym

DOKOŃCZENIE STR. 19

Didot I RETOUCHE W POLSCE (3)

Przy pomocy DIDOT-a PROFESSIONAL COLOR można tworzyć i zmieniać graficzne obiekty.

Jednakże dla grafika pracującego dotychczas piórkiem kreślarskim lub programem „pikslowym", tworzenie takiej grafiki może okazać się niezwykłym. Najpierw tworzy się zarysy elementów zorientowane obiektowo, przyporządkowuje się nim szerokość linii, kolor linii i jej typ (punktowa, kreskowa, ciągła etc.). Zamknięta płaszczyzna otrzymuje kolor wypełniający lub wypełniający wzór (w tym miejscu może być przyporządkowana cecha przezroczystości).

Można w ten sposób tworzyć szalenie ciekawe obiekty. Np. można zeskanować dowolne zdjęcie kolorowe, narysować własną figurę a następnie „wlać" w nią zeskanowany obraz. W ten sposób otrzymamy własny rysunek zamalowany w kolorze zeskanowanego obiektu.

Ponieważ DIDOT posiada dwa niezależne edytory, przy czym jeden z nich zachowuje się jak edytor grafiki wektorowej, można (wybierając dowolny krój pisma) stworzyć napis w „kolorze" zeskanowanego rysunku. Aby wyprowadzić ścieżkę wektorową, powinno się nieco więcej dowiedzieć o budowie tych obiektów. Ścieżka może składać się z segmentów prostych i krzywych. By rysować segmenty proste, ustala się punkty: początkowy i końcowy.

DIDOT rysuje ich połączenie samoczynnie. Segmenty krzywe wprowadzane są jako krzywe zdobnicze. Program oblicza przy podanych punktach krańcowych dodatkowo punkty oparcia (styczne). Wykonanie właściwego łuku jest bardzo proste (dzięki bogatemu zestawowi niezbędnych opcji). Przebieg krzywych zostanie automatycznie obliczony. Łatwo taką krzywą dodatkowo skorygować gdyż każdy z punktów oparcia może być myszą schwycony i odpowiednio przesunięty.

By umożliwić dokładną lokalizację, DIDOT proponuje mnóstwo sposobów łapania obiektów. Do dyspozycji mamy sieć magnesów, która może mieć formę siatki z dowolnym odstępem. Okręgi, linie pomocnicze mogą przejmować funkcje magnesów. Dodatkowy „promień łapania" określa siłę przyciągania takiego magnesu. W związku z tym punkty, jeśli znajdą się w polu oddziaływania magnesu, zostaną „przyciągnięte" we właściwe miejsce.

Ciekawą własnością DIDOTA jest również „odrysowywanie" obrazów pikslowych. Wystarczy obraz wczytać w jednym z dostępnych formatów (IMG, PIC) a następnie łagodnymi łukami i opcjonalnymi przejściami nastąpi wektoryzacja grafiki rastrowej. W wersji 4.143 AutoTracer (BezierTracer) jest znacznie rozbudowany dając wierne „odbitki" rysunków w formie wektorowej.

Zaletą takiego układu jest możliwość dowolnego zmniejszania lub powiększania, obrotu etc. rysunku bez straty na jakości. Można oczywiście również zamienić rysunek wektorowy na rastrowy (w formatach IMG, TIF, TIC).

DIDOT PROFESSIONAL COLOR w wersjach od 4.141 staje się podstawowym narzędziem stosowanym w poligrafii.

W technice druku używa się trzech podstawowych barw: Cyan, Magenta i Yellow z których można stworzyć w pełni kolorowy obraz. Dla druku offsetowego dany dokument można rozbić na każdy kolor oddzielnie. DIDOT wysyła przeliczone obiekty na naświetlarkę otrzymując w ten sposób wyciągi barwne. Jeśli zaistnieje potrzeba otrzymania dodatkowo barwy czarnej (po co mamy otrzymywać ją z połączenia kolorów CMY), DIDOT posiada jeszcze możliwość otrzymania czwartego, czarnego wyciągu. Otrzymujemy w ten sposób wyciąg czterobarwny (CMYK).

W tym miejscu należy dokładniej sprecyzować istotę separacji barwnej od strony praktycznej. Z połączenia barw CMY nie otrzymalibyśmy rzeczywistej czerni. DIDOT PROFESSIONAL COLOR (mowa o wersjach najnowszych) może wykorzystywać trzy metody obliczania czterech wyciągów kolorów. Każdy podstawowy kolor drukowany jest zgodnie z jego udziałem. W miejscach, gdzie występuje każdy z kolorów podstawowych w 100%, drukuje się dodatkowo czerń. Zdarza się więc, że w pewnych miejscach muszą być drukowane cztery kolory jeden nad drugim. Sprawia to techniczne problemy, gdyż często papier nie może przyjąć tyle farby.

DIDOT rozwiązuje ten problem bez zarzutu:

1) barwne kolory nie są drukowane w 100% (redukcja podkoloru)

2) dodatkowo wykorzystuje się barwę czarną, która zostaje dopasowana do zredukowanych barw. DIDOT bez zarzutów potrafi przewidzieć wynik, dzięki zachowanej w pamięci „logice" kolorów. Np. kolor szary występuje, gdy wszystkie trzy kolory występują w równych częściach.

W tej sytuacji można ułożyć następujące równanie: 20% Cyan + 30% Magenta + 40% Yellow = 20% Czerni + 10% Magenta + 20% Yellow. By drukować ten kolor, okazuje się że można całkowicie zrezygnować z Cyanu, a pozostałe kolory również się redukują. Jakie mamy z tego korzyści. Otórz łatwo jest utrzymać stabilną równowagę szarości, potrzeba mniej farby, oraz minimalizują się problemy wynikające z przyjmowaniem farby przez papier.

Wadą zaś takiego układu jest fakt, że czerń druku w rzeczywistości jest ciemnoszara, więc do końca trudno jest przewidzieć ostateczny rezultat. Dlatego też idealne efekty otrzymać można poprzez współpracę z doświadczonym operatorem maszyny offsetowej. Aby w pełni zadowolić każdego użytkownika, DIDOT posiada również bibliotekę dodatkowych kolorów zdobniczych zgodnie z obowiązującymi normami dostępnych farb.

Ogólnie mówiąc można tworzyć obiekty „pod" kolor dostępnych przez nas farb. W końcowym efekcie, jeśli wykorzystamy wszystkie możliwości separacji barw, możemy otrzymać (w wyniku naświetlenia) rozbicie obrazu nie na 4 podstawowe barwy, lecz na 4+8=12 barw !

Przy tworzeniu obiektów możemy dodatkowo ustalać kontrastowość. Na drodze od obrazu oryginalnego do wydruku na papierze występuje kilka etapów, które mogą prowadzić do zafałszowania koloru: podczas skanowania obrazu, podczas naświetlania, przenoszenia filmu na płytę drukarską oraz podczas samego druku. Tego typu rozbieżności można wykryć odpowiednimi testami, min dezyntometrem a następnie skorygować w DIDOCIE min. poprzez zmianę kontrastowości.

Kontrastowość można wprowadzić pod postacią krzywej utworzonej matematycznie a następnie stosować dla dowolnych obiektów. Bardzo ciekawymi elementami programu są: siatki oraz kalkulator. Pierwszy z nich stosowany jest do zniekształcania obiektów (teksty, rysunki). Można zniekształcać całe rysunki lub jego fragmenty (np. wyciągać uszy, nos etc).

Kalkulatorem możemy wybrać dowolny wzór, który pozwoli nam matematycznie zniekształcić obiekt. DIDOT posiada gotową bibliotekę ciekawych wzorów, daje również możliwość własnego, dowolnego definiowania. Bardzo interesującą funkcją jest możliwość formowania tekstów po dowolnej, przez nas zdefiniowanej krzywej. Wystarczy tylko myszką utworzyć dowolną krzywą a następnie „wpisać" w nią tekst. Nieco więcej na ten temat będzie w kolejnym odcinku.

Nie będę podawał (w przeciwieństwie do niektórych), że opisywany przeze mnie program jest najlepszy w dziedzinie DTP.

Na Atari w przeciwieństwie do wielu innych typów komputerów pojawiło się kilka naprawdę znakomitych produktów DTP. Każdy nich wykorzystuję do innych zadań.

Siła DIDOTA PROFESSIONAL COLOR leży w komponowaniu dokumentów (czasopisma, foldery reklamowe), gdzie powinny znajdować się obok tekstów również rysunki, kolorowe zdjęcia. Jest niezawodnym narzędziem przy prawidłowej separacji barwnej, daje wyjątkowo krótkie czasy naświetlania, poza tym zestaw: oprogramowanie, komputer, naświetlarka stanowią wyjątkowo jak na te możliwości cenę.

Jego zalety można byłoby jeszcze długo wymieniać. Używając programu do złożonych prac jestem w pełni zadowolony z uzyskiwanych efektów. Dobrze sobie radzi nawet z bardzo złożonymi problemami, przy czym jest bardzo wygodny w użyciu.

Edward Malinowski
dystrybutor produktów f-my: ***3K***

ERRATA – DOKOŃCZENIE Z POPRZEDNIEGO NUMERU

W DIDOCIE można dokonać dalszej obróbki zdjęcia zeskanowanego i opracowanego w litograficznym systemie barwnym RETOUCHE PROFESSIONAL CD, min. powiększanie/pomniejszanie, obrót zdjęcia o dowolny kąt, wykrawanie, zniekształcanie fragmentów obrazu itp.

Obrazy mogą być przenoszone do DIDOTA w jednym z formatów: TIH (obraz półtonowy), TIC (obraz pełnobarwny spośród 16,7 mln. kolorów), TIF, IMG, TIM. W przypadku innych formatów, można dokonać konwersji na format wyżej przedstawiony. By móc szybko i bez straty na jakości przedstawić linie, barwne płaszczyzny oraz dowolne kombinacje tych elementów, można wykorzystać bogate narzędzia do tworzenia rysunków wektorowych.

Treść obrazu nie jest w tym przypadku przedstawiana za pomocą konkretnych piksli, lecz matematycznymi opisami elementów obrazu i jego położenie na płaszczyźnie rysunku. W formacie wektorowym system zna wyłącznie informację o położeniu punktu początkowego oraz końcowego. Po przesunięciu obu punktów, linia zostanie od nowa kompletnie obliczona i optymalnie między obu punktami wykonana.

Zintegrowany w DIDOCIE edytor graficzny jest – podobnie jak edytor fontów – zorientowany wektorowo. Ponadto może odczytywać i zapisywać w formatach: GEM, CVG, RVF. Obiekty z dowolnych źródeł mogą zostać przez DIDOT PROFESSIONAL zarchiwizowane dla późniejszego wykorzystania. Program zarządza automatycznie własną biblioteką. Można z niej wybrany element przekopiować na aktualną stronę. Ponadto możliwe jest również założenie własnych bibliotek i zarządzać np. archiwami specyficznymi klienta.

Do składu bieżącego tekstu opracowano dwa niezależne edytory tekstu. W wersjach DIDOT 4.141 i 4.143 edytory te są tak mocno rozbudowane, że można DIDOTA używać do pełnostronicowego składu gazety bez konieczności wychodzenia na zewnątrz. Oczywiście istnieje również możliwość importu tekstu w formacie ASCII.

Pierwszym krokiem podczas opracowywania tekstu jest stworzenie szpalty. Szpalta ta zawiera informacje (oczywiście z możliwością zmian) o formacie tekstu, stylu, geometrii szpalty, rodzaju pisma, wielkości, interlinii, trwałości rejestru, justowaniu itp. Po wprowadzeniu właściwych danych, wywołany zostaje zintegrowany edytor i tekst można teraz jak na maszynie do pisania wprowadzić z klawiatury.

Jeśli formatowanie ma odbiegać od uprzednich danych dla szpalty, wówczas zostaje zamarkowany odpowiedni fragment i wywołane przynależne mu makro z informacjami o tekście. Można dokonać teraz zmian i tak np. można z łatwością dokonać podkreśleń w wierszu przez zmianę pisma, wielkości, kursywę itp. DIDOT posiada również możliwość algorytmicznego dzielenia sylab, opartego na systemie TeX (czyt. Tech). Istnieje obecnie w opracowaniu moduł dzielenia wyrazów po polsku.

Gotową szpaltę można następnie przesuwać, wycinać w dowolnym momencie nożyczkami, obracać podając odpowiednie parametry lub w sposób dynamiczny myszką. Wówczas można ustalić w dowolnym miejscu oś obrotu i przy pomocy myszki obrócić całą stronę lub jej dowolny fragment o dowolny kąt.

Śledząc od kilku lat rozwój programu DIDOT PROFESSIONAL, mogę stwierdzić, że rozrósł się w potężny pakiet DTP, potrafiący poradzić sobie ze złożonymi problemami bez konieczności korzystania z innych programów. Przykładem może tu być edytor tekstu, który w wersjach wcześniejszych nie był mocną stroną programu. obecnie należy do moich ulubionych ze względu na możliwości, prostotę obsługi, wygodę w pracy.

Istnieje jeszcze drugi rodzaj edytora. Tzw. edytor graficzny, gdzie wprowadzony tekst traktowany jest jako grafika wektorowa i w sposób dynamiczny może być podobnie jak rysunek przetwarzany.

W następnym odcinku chciałbym rozpocząć szczegółowy opis programu ze szczególnym zwróceniem uwagi na pozycje niedostępne we wcześniejszych wersjach.

Edward Malinowski

YAMAHA YM 2149

ATARI ST z pewnością nie znajduje się w czołówce komputerów domowych pod względem możliwości dźwiękowych jakie nam oferuje. Często jesteśmy narażeni na obraźliwe stwierdzenia Amigowców typu „Ale ten wasz złom stęka ..." Jednak warto wiedzieć co tak na prawdę kryje się tam pod obudową?

Chyba wszyscy już słyszeliśmy o tak bardzo rozsławionym przez spectrum-owców AY 3-8190. Znajdujący się w ATARI ST układ YAMAHY YM 2149 jest bardzo zbliżonym do niego programowalnym generatorem dźwięku, który znalazł zastosowanie w wielu automatach gier zręcznościowych. Ta 40-nóżkowa kość oferuje nam 3 niezależnie programowane generatory dźwięku (A,B,C), generator szumu, programowany mieszacz dźwięku oraz 2 równoległe 8-bitowe porty. YM zawiera 16 rejestrów jednobajtowych umożliwiających tworzenie i obróbkę dźwięku. Równoległe porty służą do komunikacji z urządzeniami zewnętrznymi (np.drukarką).

W poszczególnych rejestrach ustalamy częstotliwość dźwięku, kształt obwiedni, głośność, czasy wybrzmiewania oraz tryb pracy generatorów. Komputer pamięta zawartość rejestrów do momentu wpisania do nich innych wartości, dlatego nie jest konieczne odnawianie wszystkich parametrów generatora przed odtworzeniem dźwięku. Zapoznajmy się z przeznaczeniem rejestrów:

Rejestry 0 i 1 : służą do określenia częstotliwości dźwięku dla kanału A. Wykorzystywany jest cały rejestr 0 i 4 najniższe bity rejestru 1.

Rejestry 2 i 3 : jak wyżej, lecz dla kanału B.

Rejestry 4 i 5 : jak wyżej, lecz dla kanału C.

Rejestr 6 : 5 najniższych bitów tego rejestru określa czestotliwość generatora szumu.

Rejestr 7 : Rejestr kontrolny mieszacza dźwięku,

Bit 0 - kanał A dla dźwięku 0 - wł. 1 - wył.
Bit 1 - kanał B dla dźwięku 0 - wł. 1 - wył.
Bit 2 - kanał C dla dźwięku 0 - wł. 1 - wył.
Bit 3 - kanał A dla szumu 0 - wł. 1 - wył.
Bit 4 - kanał B dla szumu 0 - wł. 1 - wył.
Bit 5 - lanał C dla szumu 0 - wł. 1 - wył.
Bit 6 - port A jako : 0 - wejściowy 1 - wyjściowy
Bit 7 - port B jako : 0 - wejściowy 1 - wyjściowy

Rejestry 8 9 10: określają głośność kanałów A B C. Wartość (0-15)

Rejestry 11 12 : zawartość rejestru 11 to młodszy bajt fazy wybrzmiewania, 12 to starszy.

Rejestr 13 : określenie kształtu obwiedni. Wykorzystuje się bity 0-3. Zależność przedstawiona na rysunku.

Rejestry 14 15 : odpowiednio jest to port A i port B. Zazwyczaj port A używany jest przez stację dysków, port B służy do współpracy z drukarką.

YM jest adresowany pośrednio. W pamięci RAM znajdują się 2 rejestry służące do programowania tego układu. Są one umieszczone pod adresami $FF8800 i $FF8802. Pierwszy z nich zwany jest rejestrem odczytu danych i selekcji ($FF8800). $FF8802 nosi nazwę rejestru zapisu danych. Chcąc zmienić zawartość jednego z 16 rejestrów YAMAHY, należy najpierw umieścić jego numer w rejestrze selekcji, a następnie do rejestru danych wpisać nową wartość. Jak wskazuje sama nazwa, rejestr selekcji i odczytu danych służy również do odczytu danych. Jeżeli ostatnio adresowany był port A, to odczytując $FF8800 otrzymamy nie liczbę 14 (port A), lecz zestaw informacji dotyczacy stacji dysków (numer stacji, rodzaj stacji). Należy tu wspomnieć, że oba rejestry $FF8800 i 02 mogą zostać zmienione tylko w trybie nadzorcy (SUPERVISOR), a próba ich zmiany w trybie użytkownika skończy się błędem szyny BUS ERROR (2 bomby). Zajmijmy się przez chwilę GFA basic-iem. Istnieją tu oczywiście gotowe komendy do tworzenia dźwięku. Pierwsza z nich to :

SOUND kanał(1-3), głośność(0-15), nuta(1-12), oktawa(1-10)

Jak widać instrukcja ta odgrywa daną nutę, lecz nie ma żadnego wpływu na ustawienie barwy generowanego dźwięku. Robi to komenda WAVE. Zainteresowanych odsyłam do książki GFA-BASIC. W tym momencie każdy stwierdzi, że takie programowanie muzyki jest bardzo żmudne, ...i ma rację. Konieczna jest tu mała dygresja odnośnie ROM-u ST. Znajdują się tam biblioteki procedur wspomagających wykorzystanie sprzętowych możliwości komputera. Jedną z nich jest XBIOS, umieszczone są w niej procedury ułatwiające obsługę YAMAHY. Są to odpowiednio Glacess o numerze 28 i Dosound o numerze 32. Przykładowy sposób wywołania ich z poziomu asemblera i GFA jest przedstawiony na dołączonym listingu. Glacess automatycznie ustawia zadane wartości w odpowiednich rejestrach, umożliwia również odczytanie dowolnego rejestru. Dosound to procedura odgrywająca muzykę w formacie X32, działa ona niezależnie od głównego programu. X32 składa się z ciągu poleceń i koniecznych im danych.

Polecenia $00 - $0F interpretowane są jako wpisanie do określonego przez numer polecenia rejestru YM, 1-bajtowej wartości następującej po poleceniu.

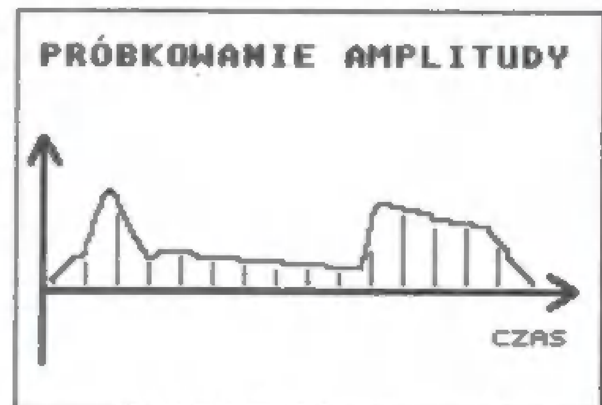
Polecenie $80 - Bajt po tym poleceniu ładowany jest do rejstru tymczasowego. (1-bajtowy pomocniczy rejestr YM umieszczony w RAM)

Polecenie $81 - Po tym polecenu występują 3 bajty danych. Pierwszy określa numer rejestru do którego zostanie wpisana zawartość rejestru tymczasowego. Drugi wartość, która będzie systematycznie dodawana do rejestru tymczasowego. Trzeci zawiera wartość końcową. Cała operacja kończy się gdy wartość w rejestrze tymczasowym równa jest wartości końcowej. Procedura ta działa na zasadzie pętli.

Polecenia $82 do $FF - następujący po nich bajt danych określa czas oczekiwania do pobrania kolejnego polecenia. Wyrażony on jest liczbą tyknięć zegara 50 Hz. Jeżeli przyjmuje wartość 0, wtedy następuje zakończenie wykonywania procedury Dosound.

Istnieje kilka programów, które pozwalają tworzyć utwory w zapisie nutowym, a następnie zapisywać je w formacie X32 (np. XBIOSMSK). Jednak tego typu muzyka odgrywana na 50 Hz czy 200 Hz jest bardzo daleka od wymarzonego czystego brzmienia jakie chcielibyśmy uzyskać na naszym komputerze. Rzeczywiście wierne odtworzenie brzmień instrumentów możemy otrzymać poprzez ich digitalizację. Spójrzmy na rysunek. Każdy dźwięk można przedstawić jako zmieniającą się w funkcji czasu amplitudę. Pobierając próbki (wartości amplitudy) w odpowiednich, rów

nych odstępach czasu, jesteśmy w stanie na ich podstawie odtworzyć kształt przebiegu amplitudy w czasie. Służy do tego urządzenie zwane samplerem lub digitizerem dźwięku. Im mniejsze będą odstępy czasu między kolejnymi próbkami (większa częstotliwość próbkowania), tym otrzymany przebieg będzie wierniejszy rzeczywistemu.



Chcąc odtworzyć dźwięk wystarczy z odpowiednią częstotliwością wpisywać do rejestrów YAMAHY, odpowiedzialnych za amplitudę, wartości próbek. W praktyce należy jeszcze przekształcić te próbki, ze względu na nieliniowość wzmacniacza YAMAHY. Zostanie to dokładnie omówione w następnym numerze ST- FA-Na. Metoda ta daje niesamowite efekty, zresztą na podobnej zasadzie działa odtwarzacz płyt kompaktowych, jest ona jednak bardzo pamięciochłonna.

REJESTR OBWIEDNI

BITY 3	2	1	0	
0	0	0	0	
0	1	0	0	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Zazwyczaj próbki są 8-bitowe, co umożliwia reprezentację 256 poziomów amplitudy. Łatwo wyliczyć, że jedna sekunda muzyki próbkowanej z częstotliwością 16 KHz zajmuje 16 Kb pamięci. Prawdziwie czysty dźwięk możemy uzyskać przy 16 KHz, czasem nawet dopiero powyżej 20 KHz. Odgrywanie z takimi dużymi częstotliwościami zajmuje często znakomitą większość czasu naszego procesora, a to już jest zły objaw.

W ATARI STE zastosowano nowy układ pozwalający na odtwarzanie stereofonicznej digitalizacji aż do 100 KHz, ...i to (sic!) bez uczestnictwa procesora. Dodatkowo posiada on rejestry kontrolujące natężenie tonów niskich, średnich, wysokich oraz balansu między lewym i prawym kanałem. Istny raj dla muzyków.

GRAYHOUND

Przykładowy program wywołujący procedurę Dosound z poziomu asemblera (DEVPACK): (Wpisywać bez komentarzy)

```
clr.w -(sp) -czyść stos
move.l #dane,-(sp) -wskaźnik danych muzyki
move.w #32,-(sp) -numer procedury
trap #14 -wywołanie
XBIOS add.w #8,sp -wyrównanie stosu
dane: dc.b 0,0,1,0,2,0,3,0,4,0,5,0,6,0,7,0
Wywołanie Dosound z poziomu GFA:
Ładowanie pliku zawierającego muzykę w X32:
muzyka$=SPACE$(32000) -rezerwacja pamięci
dane%=VARPTR(muzyka$) -wskaźnik do danych
BLOAD "*.X32",dane% -ładowynie muzyki
VOID XBIOS(32,L:dane%) -wywołanie Dosound
Wywołanie Giacess z poziomu asemblera:
W miejsca rejestr i dana należy wpisać odpowiednie wartości,
clr.w -(sp) -czyść stos
move.w #rejestr,-(sp) -numer rejestru
move.w #dana,-(sp) -wartość
move.w #28,-(sp) -numer procedury
trap #14 -wywołanie
XBIOS add.w #8,sp -wyrównanie stosu
Wywołanie procedury Giacess z poziomu GFA;
rejestr%=5
dana%=4
VOID XBIOS(28,dana%,rejestr%)
```

```
Przykład pośredniego adresowania rejestrów YAMAHY
z poziomu asemblera:
*wejście w tryb nadzorcy
clr.w -(sp)
move.w #$20,-(sp)
trap #1 addq #4,sp
*załadowanie rejestrów
*w miejsce rejestr i dana wpisujemy wartości
move.b #rejestr,$FF8800
move.b #dana,$FF8802
*szybkie wyjście z programu
clr.l -(sp) trap #1
```


Ta część kartki została wycięta.
Prawdopodobnie był tu formularz prenumeraty.

Z pośród wielu programów MIDI, szczególnie duże zainteresowanie wzbudzają edytory MIDI. Tworzą one rodzinę programów, które mają za zadanie ułatwić programowanie nowych brzmień na syntezatorach. Przyjemne i łatwe w obsłudze graficzne środowisko programów w dużym stopniu pomaga w tym trudnym dziele. Dodatkowa wizualizacja graficzna danych, duża moc obliczeniowa komputera i zdecydowanie większa niż w syntezatorze pamięć operacyjna to kolejne atuty.

Pozostaje tylko drobny szczegół o którym wspomnieliśmy w jednym z artykułów o MIDI. Dane dotyczące brzmienia w syntezatorze są zrozumiałe jedynie dla konkretnego modelu. Tak więc tylko bardziej znane „klawiatury" otrzymały swoich cyfrowych braci do pomocy. Zaszczytu tego doczekał się również szeroko osławiony model YAMAHY, DX-7. Wersja edytora na ATARI ST nosi nazwę:

DXPERT

i została wydana przez firmę FIVE PIN DIN w roku 1987.

DXpert wita nas miłym dla oka, niezbyt skomplikowanym ekranem, który niestety w posiadanej przez nas wersji uruchamia się jedynie na monitorze mono. Wbrew pierwszemu wrażeniu, program zawiera bardzo dużą liczbę opcji. Główne biurko podzielone jest na 3 bloki. Pierwsze dwa są to banki głosów A i B, trzeci reprezentuje bufor, w którym przechowywane są aktualnie obrabiane brzmienia.

Desk File Send Receive Options Randomize DXpert by FIVE_PIN_DIN ©1987

BANK A - VARIOUSS				BANK B - TEST				BUFFERS	
1	STRINGS M1	17	FULL SYNTH	1	MOD-SYNTIX	17	MOD-SYNTIX	1	MOD-SYNTIX
2	GLASS	18	E FILTER 1	2	SYNTH JH	18	POLYMOOG 2	2	
3	JUNOSTRING	19	STAR RISE	3	PERC SYN A	19	CHOIRSYN	3	
4	STRM1 LNGA	20	WISH MALE	4	SYNTHYX	20	MOOG 1	4	
5	STRING2S-H	21	FILTSPLIT2	5	SYNTHY-1	21	LEAD-1JH	5	
6	BELL-STRGM	22	MULTISOUND	6	DIGI A	22	PITCH SYNT	6	
7	NEOPIANORC	23	CLAV-FLUTE	7	DIGI B	23	DIGIX	7	
8	FILTRBRASS	24	POWER-CLAV	8	BIWA	24	SYNTH.....	8	
9	BUZZPERCUS	25	VOICE M1	9	DIGI D	25	FROGGY		
10	ELECTROCEL	26	PAN PIPES	10	DIGI E	26	LEAD-2JH		
11	PSYCOSPACE	27	D.B--WAHH	11	SYNTH A	27	OBERIX	MIDI CHANNEL	
12	MYSTERY555	28	BIRDSONG	12	SYNTH B	28	CHICK SYNT	◇ 01 ◇	
13	FENDERMODE	29	SQUARE SY1	13	SYNTH C	29	SYNTHIX		
14	VCF SWEEP	30	GRANDPIANO	14	SYNBELLS A	30	SYNTY-WAH		
15	LFO SWEEP	31	SYNTH CLAV	15	SYNVOICE A	31	LORD-JH		
16	SAW SYNTH	32	ORGANBRASS	16	BELL A	32	ORCH.JH		

Dwa takie same banki po 32 głosy znajdują się w pamięci syntezatora. DXpert jako w pełni 'rasowy' edytor pozwala nam dokonywać wszelkiego rodzaju operacje na danych MIDI dotyczących barw. Każdy głos posiada swoją nazwę oraz numer. Przemieszczanie głosów jest podobne do przemieszczania ikon plików. Wystarczy najechać na wybraną nazwę, i ... zanieść ją gdzie trzeba. Mając instrument podłączony do komputera przewodami MIDI możemy bezpośrednio skopiować zawartość całego banku, czyli 32 głosów z syntezatora do kompuera jak i odwrotnie. Często chodzi nam jednak o edycję parametrów pojedyńczego brzmienia. Z poziomu instrumentu wszystkie potrzebne operacje są możliwe do wykonania, jednak w praktyce wymaga to dużej wprawy i dobrej znajomości algorytmów generowania głosów.

Mając do dyspozycji powyższy program wszystko możemy uzyskać poruszając się po przejżystym menu, bez potrzeby zagłebiania się w tajniki instrumentu. Wybrany głos przenosimy do jednej z ośmiu pozycji bufora, a do planszy edycji przechodzimy opcją EDIT... z menu OPTIONS. Jeden z rysunków przedstawia kopię ekranu op wykonaniu tej operacji. W DX-7 brzmienie powstaje w wyniku wzajemnej modulacji 6 niezależnych generatorów. Daje to w efekcie prawie nieograniczone możliwości.

Każdy z generatorów może tworzyć przebieg jednej składowej, może również być modulowany przez inny generator w czasie rzeczywistym. Generatory pracują w jednym z 18 algorytmów zawierających liczne pętle sprzężeń zwrotnych, opóźnień itd... Tak, po krótce można opisać powstawanie brzmienia. Wszystkie parametry generatorów znajdują się na ekranie, pogrupowane w bloki tematyczne.

Desk File Send Receive Options Randomize Editing Voice: MOD-SYNTIX

ENVELOPES Draw

OP	Rate 1	2	3	4	Level 1	2	3	4
6	60	15	15	70	99	98	88	0
5	60	20	20	70	99	98	88	0
4	60	20	20	70	99	98	88	0
3	60	20	20	79	99	98	88	0
2	13	20	20	34	99	98	88	0
1	99	99	99	99	99	99	99	0
	94	67	95	60	50	50	50	50

PITCH ENVELOPE

FREQUENCY Draw

OP	Fine	Coarse	Det	Mod
6	39	01.38	0	FR
5	00	02.00	0	FR
4	00	00.50	0	FR
3	00	00.50	0	FR
2	00	01.00	+7	FR
1	00	00.50	0	FR

KEYBOARD LEVEL SCALING Draw

OP	Left Curve	Dp	Right Curve	Dp	Break Point	Rte Scl
6	-LIN	0	-LIN	0	A -1	2
5	-LIN	50	-LIN	20	C 4	2
4	-LIN	50	-LIN	0	C 4	2
3	-LIN	0	-LIN	0	A -1	2
2	-LIN	0	-LIN	0	C 5	2
1	-LIN	0	-LIN	0	A -1	2

KEY SYNC ON T'SPOSE C 3
VOICE NAME MOD-SYNTIX
PLAY Note C 3
Volume 127 Duration 25

OP6	OP5	OP4	OP3	OP2	OP1
ON	ON	ON	ON	ON	ON

LFO
Wave SAW DOWN
Speed 33
Delay 00
PMD 0
AMD 99
PMS 3
Sync ON

LEVELS

OP	Out Lev	Vl Sn	Md Sn
6	0	2	0
5	89	2	0
4	85	0	0
3	99	1	3
2	99	1	2
1	98	0	0

ALGORITHM
Number 18
Feedback 5

COPY
UNDO
SEND EDITS OFF
SEND VOICE
END

Zmian dokonujemy tupiąc mysza na wybranych wartościach. Możemy również graficznie ustalić charakterystyczne parametry przebiegu. Obwiednia, częstotliwość, dokładne dostrojenie klawiatury, poziomy głośności poszczególnych sygnałów. Włączanie lub wyłączanie generatorów (zwanych operatorami) przeskalowanie dźwięku oraz graficzna ikona algorytmu użytego znajdują się w dolnej części ekranu.

Możemy również wysłać głos do syntezatora opcją SEND VOICE. Dokonywane na bierząco zmiany bez problemu są odsłuchiwane opcją PLAY w lewym dolnym rogu. Proponuję poeksperymentować. Powróćmy do Menu głównego. Informacje o głosach przesyłane są na jednym z 16 kanałów MIDI, do sprecyzowania tego służy blok z napisem MIDI CHANNEL. Bardzo interesującą opcją programu jest możliwość losowego generowania banku 32 głosów według podanych wyznacznych.

Program współpracuje z drukarką, posiada aktywny help. Przyjżyjmy się teraz poszczególnym opcją z górnej lini menu.

DOKOŃCZENIE GFA BASIC

wykonaniu polecenia NEXT, licznik zostaje zwiększony o 1. Będzie on zwiększany dopóty, dopóki nie osiągnie wartości końcowej (w tym przypadku 10). Wtedy pętla zostanie opuszczona. Należy pamiętać, że po słowie NEXT musi być zawsze umieszczona nazwa licznika, który ma być inkrementowany o 1 (zwiększany co 1). Aby upewnić się, że ta "teoria" jast słuszna, proszę wpisać poniższy program, który wyświetla zawartość licznika, podczas pracy pętli.

```
For X=1 To 10
Print "licznik = ";X
Next X
```

GFA BASIC umożliwia przy stosowaniu tego rodzaju pętli, odliczanie co 1 w dół, a także w górę z określonym krokiem. Aby licznik pętli zmniejszany był co 1, zamiast "TO", należy wpisać "DOWNTO". UWAGA: Wartość początkowa licznika musi być większa od końcowej.

```
For A=10 Downto 1
Print "licznik = ";A
Next A
```

Licznik Pętli nie musi być zwiększany co 1 ale np.: co 0.5; 2; 0.12; 5; itd. Do tego celu, po słowie TO wpisujemy STEP, oraz krok z jakim inkrementowany będzie licznik. Dla przykładu prześledźmy "zachowanie" się licznika w poniższym programie.

```
For X=1 To 10 Step 0.5
Print "licznik = ";X
Next X
```

REPEAT...UNTIL

Ciąg instrukcji zawartych między słowami REPEAT, a UNTIL powtarzany jest dotąd, aż warunek umieszczony na końcu pętli, po słowie UNTIL, zostanie spełniony. W przeciwieństwie do pętli FOR...NEXT, liczba powtórzeń, nie jest z góry określona.

```
Repeat
Input "Wprowadz liczbe ";A
Until A=0
Print "Koniec pętli"
```

Dopóki liczba wprowadzona z klawiatury będzie różna od 0, pętla będzie cały czas powtarzana.

DO...LOOP

Jest to tzw. "pętla bez wyjścia". Ciąg instrukcji umieszczony między słowami "Do" i "Loop" wykonywany jest przez cały czas, np.:

```
Do
Inc X
Print X
Pause 10
Cls
Loop
```

WHILE...WEND

Ciąg instrukcji umieszczony między WHILE i WEND będzie powtarzany, dopóki spełniony będzie warunek umieszczony na początku pętli, tuż po słowie WHILE, np.:

```
A=10
While A>0
Dec A
Print A
Wend
```

W GFA BASICU istnieje jeszcze komenda, która umożliwia opuszczenie pętli FOR...NEXT i DO...LOOP, w przypadku spełnienia warunku podanego jako jej argument (jest to komenda EXIT), np.:

```
X=0
Do
Inc X
Print X
Exit If X=10
Loop
Print "Koniec"
```

Opanowanie dzisiejszego tematu w stopniu dostatecznym, pozwoli Państwu w późniejszym terminie (a nawet już teraz), budować skomplikowane i coraz ciekawsze programy. Na koniec lekcji, jak zawsze przykładowy program.

```
Input "Ile linii ";Il%
Cls
For Y%=1 To Il%
For X%=1 To 640 Step Y%
Plot 640-X%,Y%
Plot X%,Y%
Plot 640-X%,2*Il%-Y%
Plot X%,2*Il%-Y%
Next X%
Next Y%
```

Codie

Literatura: [1] GFA BASIC - Stefan Nawrocki [2] TURBO PASCAL 4.0 - Andrzej Marciniak

DOKOŃCZENIE DXPERT

File	Options	Randomize
Load Active Bank	Edit Buffer Voice	Generate with following
Save Active Bank	Print Active Voice	✓ Fixed Root
Accessing Drive A	Print Active Bank	✓ Envelopes
Quit	Help	✓ Frequency
Send	**Receive**	✓ Keyboard Scaling
Send Active Bank	Request Bank	✓ Output Levels
Send Active Voice	Request Voice	✓ LFO
	Wait for Bank	✓ Large Variance (30%)
	Wait for Voice	Small Variance (10%)

File:

Load Active Bank - ładuje z dysku 32 głosy do aktywnego banku. Poprzednie barwy ulegną skasowaniu.

Save Active Bank - zapisuje bank na dysku

Accessing Drive A - wybór stacji

Quit - wyjście z programu

Send:

Send Active Bank - wysyła do syntezatora bank 32 głosów. Syntezator musi mieć odbezpieczo ną pamięć głosów wewnętrznych i oczekiwać na przyjęcie banku danych.

Send Active Voice - przesyła do syntezatora po jedyńczą barwę o zadanym numerze. Jak wyżej należy odbezpieczyć pamięć wewnętrzną i przejść do trybu oczekiwania na pojedyńczą barwę.

Receive:

Request Bank - wczytuje do komputera bank 32 gło sów wysłanych z instrumentu.

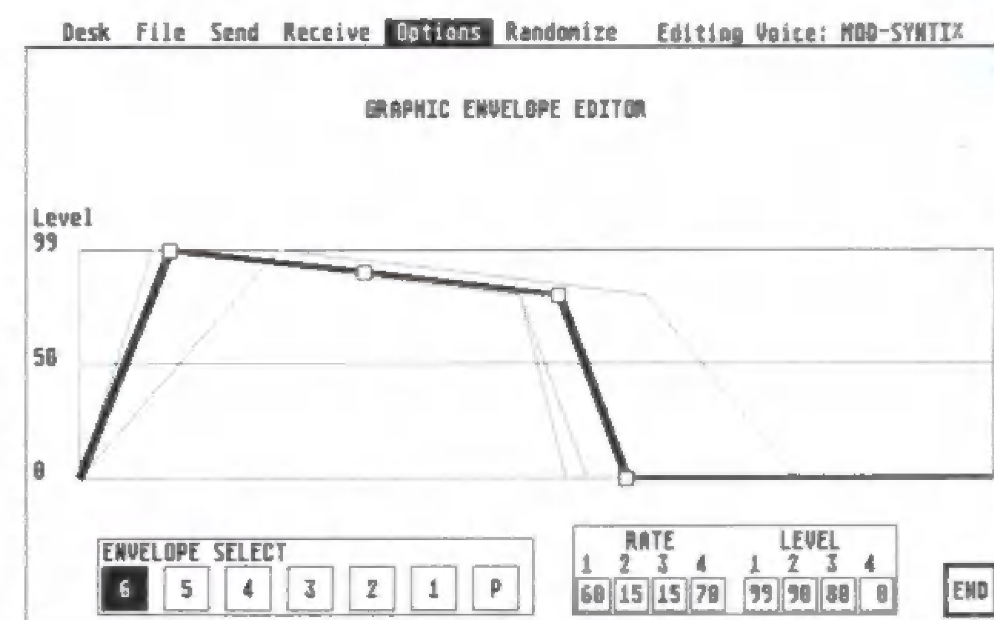
Request Voice - jak wyżej, lecz pojedyńczy głos

Wait For Bank - oczekuje na wysłanie banku da nych z instrumentu.

Wait For Voice - oczekuje na wysłanie jednego głosu do komputera

Randomize:

Generate... - generuje bank 32 losowych głosów.



DXpert nie jest jedynym powstałym edytorem dla DX-7, w pełni jednak zdaje egzamin w praktyce. W wyniku szybkiego rozwoju systemu MIDI ostatni mi laty, zostały napisane liczne programy do obrób ki brzmień instrumentów firmy ROLAND, KORG, KAWAI.

Istnieją również rozwiązania obsługi przez komputer profesjonalnych samplerów np. AKAI, jed nak przy pojemności pamięci 1MB jest to raczej cie kawostka niż narzędzie.

(GRAYHOUND)

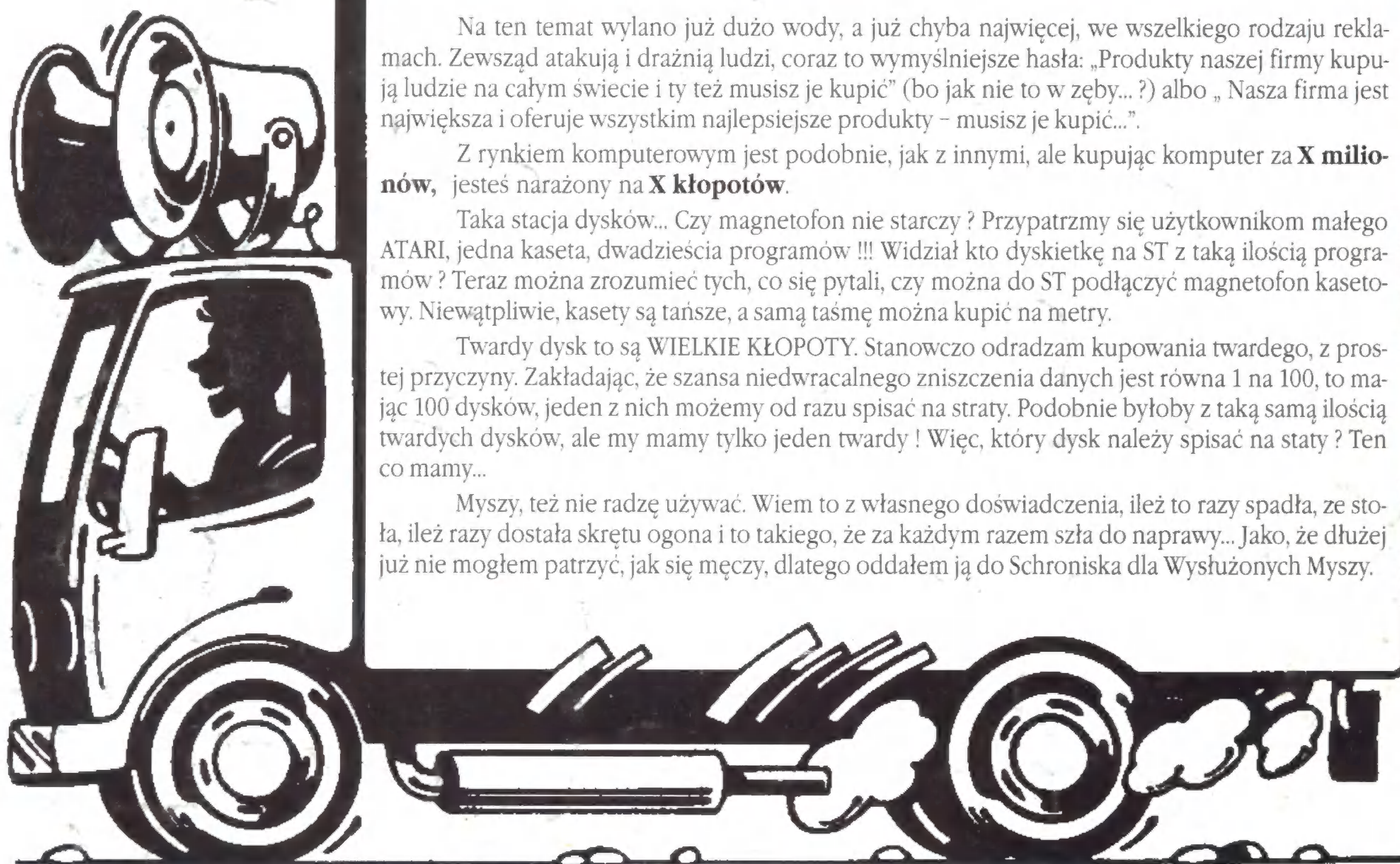
LISTA PRZEBOJÓW

Zgodnie z Waszym życzeniem od dzisiejszego numeru ST-FANa rozpoczynamy redagowanie LISTY PRZEBOJÓW gier na ATARI ST. Uznaliśmy, że Macie rację twierdząc, że żadne pismo komputerowe takiej listy nie prowadzi. Wychodzimy Wam na przeciw. Lista ta jest redagowana na podstawie ilości, najlepiej sprzedających się gier. Oczekujemy na Wasze propozycje do LISTY PRZEBOJÓW. Jednocześnie redakcja ST-FANa ogłasza konkurs na najciekawszą nazwę dla LISTY PRZEBOJÓW. Wasze propozycje proszę przesyłać na kartach pocztowych z dopiskiem KONKURS. Dla zwycięzcy nagroda-niespodzianka.

Jerry

SYMULATORY	ZRĘCZNOŚCIOWE	PRZYGODOWE	SPORTOWE
F-19 STELATH FIG.	ALCATRAZ	ELVIRA 2	SENSIBLE SOCCER
F-16 FALCON	JIM POWER	LURE OF TEMPTRESS	MICROPROSE GOLF
FORMULA ONE GP	DOUBLE DRAGON 3	AMBERSTAR	ADVANTAGE TENNIS
SILENT SERVICE 2	WINGS OF DEATH 2	A.G.E.	KICK OFF 2
F-29 RETALIATOR	WIZZKID	FASCINATION	TIP OFF

Duży może więcej ?



Na ten temat wylano już dużo wody, a już chyba najwięcej, we wszelkiego rodzaju reklamach. Zewsząd atakują i drażnią ludzi, coraz to wymyślniejsze hasła: „Produkty naszej firmy kupują ludzie na całym świecie i ty też musisz je kupić" (bo jak nie to w zęby... ?) albo „ Nasza firma jest największa i oferuje wszystkim najlepsiejsze produkty - musisz je kupić...".

Z rynkiem komputerowym jest podobnie, jak z innymi, ale kupując komputer za **X milionów,** jesteś narażony na **X kłopotów**.

Taka stacja dysków... Czy magnetofon nie starczy ? Przypatrzmy się użytkownikom małego ATARI, jedna kaseta, dwadzieścia programów !!! Widział kto dyskietkę na ST z taką ilością programów ? Teraz można zrozumieć tych, co się pytali, czy można do ST podłączyć magnetofon kasetowy. Niewątpliwie, kasety są tańsze, a samą taśmę można kupić na metry.

Twardy dysk to są WIELKIE KŁOPOTY. Stanowczo odradzam kupowania twardego, z prostej przyczyny. Zakładając, że szansa niedwracalnego zniszczenia danych jest równa 1 na 100, to mając 100 dysków, jeden z nich możemy od razu spisać na straty. Podobnie byłoby z taką samą ilością twardych dysków, ale my mamy tylko jeden twardy ! Więc, który dysk należy spisać na staty ? Ten co mamy...

Myszy, też nie radzę używać. Wiem to z własnego doświadczenia, ileż to razy spadła, ze stoła, ileż razy dostała skrętu ogona i to takiego, że za każdym razem szła do naprawy... Jako, że dłużej już nie mogłem patrzyć, jak się męczy, dlatego oddałem ją do Schroniska dla Wysłużonych Myszy.

Jakież to szczęście nie mieć monitora wysokiej rozdzielczości... Wiadomo, dużo pikseli - dużo kłopotów. Widział kto grę na ten monitor ? Jest ich chyba mniej, niż klawiszy w czterodziałaniowym kalkulatorze...

Mam cichą nadzieję, że w chwili, w której czytasz te słowa drogi Czytelniku, twój komputer znajduje się w bezpiecznej odległości, stanowczo odradzam wszelkich destruktywnych działań z Twojej strony *.

Ja swój komputer oddałem bezrobotnym, wraz z kilkoma grami. Teraz nie będą się nudzić i będą mieli, coś do roboty.

#1 P.S. Radzę dobrze zapamiętać tych kilka przykładów, obalających tezę, że DUŻY MOŻE WIĘCEJ. Można się na nie powołać, w wypadku ataków ze strony użytkowników innych komputerów, twierdzących, że ich komputery są lepsiejsze, bo mają większiejsze możliwości.

#2 P.S. Z uwagi, na to, że powyżej wymienieni użytkownicy INNYCH komputerów, mogą podjąć akcję odwetową, a sam autor tego felietonu nie chce się obudzić z joystic'kiem w plecach, dlatego pozostanę w podziemiu.

Wasz POL-ANONIM.

* Dotyczy to również darcia na strzępy tego numeru ST-Fana, pod karą pozbawienia dostępu do komputera na okres jednego roku lub w wyjątkowym przypadku, karą powieszenia na kablu od myszy, w zawieszeniu, na okres jednego tygodnia.